For Reference

NOT TO BE TAKEN FROM THIS ROOM

For Reference

NOT TO BE TAKEN FROM THIS ROOM

Ex libris
universitatis
albertaensis

QUAECUMQUE VERA

The University of Alberta
Printing Department
Edmonton, Alberta

Digitized by the Internet Archive
in 2023 with funding from
University of Alberta Library

https://archive.org/details/Wilson1970_2

THE UNIVERSITY OF ALBERTA

[illegible]

[illegible]

© [illegible]

A THESIS

SUBMITTED [illegible]

IN PARTIAL [illegible]

OF [illegible]

DEPARTMENT OF [illegible]

[illegible]

[illegible]

THE UNIVERSITY OF ALBERTA

OPTIMIZATION PROCEDURES FOR REDUCING DRILLING COSTS

BY

© DAVID C. WILSON

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE IN PETROLEUM ENGINEERING

DEPARTMENT OF CHEMICAL AND PETROLEUM ENGINEERING

EDMONTON, ALBERTA

SPRING, 1970

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "Optimization Procedures for Reducing Drilling Costs" submitted by D.C. Wilson in partial fulfilment of the requirements for the degree of Master of Science in Petroleum Engineering.

Abstract

Until recently the drilling of a well was much more of an art than a science. With the advent of mathematical models of the drilling process it has become possible to estimate quantitatively the effect of certain of the parameters involved.

The object of this thesis was to investigate various optimization procedures and how they could be used in conjunction with the mathematical model selected, in order to reduce the cost of drilling a well.

The present knowledge of the drilling process restricted the number of parameters to be optimized to two, namely, the weight on the bit and the rotary speed. Other factors, such as mud properties and bit type, were assumed to have been properly selected. Within these limits, three methods of varying complexity have been developed. The first method seeks to minimize the cost per foot drilled during a bit run, the second method minimizes the cost of a selected interval and the third method minimizes the cost over a series of intervals. The methods are listed in order of increasing complexity.

It was found that all of the methods gave a worthwhile cost saving and that the saving increased as the complexity of the method increased. The data requirements for the methods increased with increasing method complexity.

Further improvements will depend on greater sophistication of the mathematical models used to describe the drilling process.

Acknowledgements

The author wishes to express his appreciation to Dr. R.G. Bentsen for his help and guidance throughout the course of this investigation.

Thanks are due to the following:

- Mobil Oil Canada Limited for supplying the data used in testing the computer programs.
- The National Research Council of Canada and the Alberta and Southern Gas Company Limited for their financial assistance.
- Miss Wozniuk for her prompt help in typing the thesis.
- Last, but by no means least, a special appreciation is due to my wife for her encouragement and patience throughout this work, in addition to proof reading.

Table of Contents

Page

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables vii

List of Figures viii

I Introduction 1

II Review of Available Models 3

III General Theory 14

  A. Summary of the Galle and Woods Equations 14

  B. Calculation of Parameters in the Galle and Woods Equations 17

  C. The Galle and Woods Optimization Procedure 22

  D. Modified Galle and Woods Method 31

  E. "Multi-interval" Optimization 41

IV Discussion of Results 48

  A. Introduction 48

  B. "Point" Optimization 49

  C. "Interval" Optimization 50

  D. "Multi-interval" Optimization 51

  E. Comparison of the Methods 66

V Conclusions and Recommendations 69

Table of Contents (continued)

Page

Nomenclature 71

Bibliography 75

Appendix

A. Derivation of Footage Distribution

B. Program and Results of $1(\overline{W})$ Fit

C. Program and Results of Integral Fit

D. Program and Results of the Parameters Calculations

E. Mobil Oil Test Data

F. Program and Results of "Point" Optimization

G. Program and Results of "Interval" Optimization

H. Program and Results of "Multi-interval" Optimization

List of Tables

Table No. Page

1 Cost Variation with Number of Intervals Using a 5 x 5 Matrix 59

2 Cost Variation with Number of Intervals Using a 9 x 9 Matrix 60

3 Cost Variation with Number of Intervals Using a 13 x 13 Matrix 61

4 Cost Variation with Matrix Size Using 9 Intervals 62

5 Cost Variation with Matrix Size Using 13 Intervals 63

6 Cost Variation with Matrix Size Using 18 Intervals 64

7 Comparison of Different Optimization Techniques .. 67

List of Figures

Figure No.		Page
1	Cost Variation with Weight and Speed	53
2	Cost Variation Using First Order Interpolation ..	55
3	Cost Variation Using Second Order Interpolation .	56
4	Effect of Matrix Size and First Order Interpolation	57
5	Effect of Matrix Size and Second Order Interpolation	58

I. Introduction

The nature of the oil industry is such that it is always facing uncertainty. It is never certain that any particular well drilled will be an economic producer, although the probability of finding a commercial well is, of course, much higher in a developed area than it is in a wildcat situation. Because of the number of dry holes that do occur, it is necessary that the few successful finds give a large return on investment.

There are several ways that an oil company can improve its chances of achieving a reasonable rate of return on investment. The first indication that oil or gas may exist in an area comes from the application of geological and geophysical techniques, so that any improvement that is made in the interpretation of this type is helpful. To this day, however, the only way of confirming the existence of oil or gas is to drill a well. Hence it follows that the lower the cost of drilling each well the greater the number of wells that can be drilled and the greater the probability of finding producing wells.

In spite of this rather obvious fact, the drilling of wells was, until very recently, more of an art than a science. It was not until the late fifties and early sixties that the advent of mathematical descriptions of the drilling process changed the situation.

The object of this dissertation is to examine how optimization

techniques may be used in conjunction with the published models in order that the drilling cost may be minimized. It should be realized that the total cost of a well involves other factors such as the amount of testing and coring and that the return on money invested in these areas should also be considered.

Any technique suggested will give poor answers if the model describing the system is inadequate. The second chapter of this work will therefore examine the progress made to date in forming a mathematical model of the processes involved in drilling a well. Chapter III will give the mathematical background to the various optimization techniques suggested and Chapter IV will review the results obtained using each method.

In view of the fact that the models currently available may be improved upon in the future, every effort has been made to write the computer programs in a general format. This should permit, relatively easily, the substitution of any improved model in the programs.

II. Review of Available Models

Although many prior investigations had taken place into the various aspects of drilling performance, Speer (28), in 1958, was the first to propose a generalized method for determining optimum drilling techniques. Speer suggested that the drilling performance depended primarily upon five controllable factors:

1. Properties of the circulating medium.
2. The hydraulic effects.
3. Type of bit.
4. Weight on bit.
5. Rate of rotation.

Investigations since 1958 have confirmed and refined Speer's suggestions.

The properties of the circulating fluid that have been found to affect drilling are as follows:

1. Mud weight:

   It has been shown (21,13) that drilling rate decreases as the mud pressure increases, and that the decrease is actually more correctly attributed to the excess of the hydrostatic pressure over the formation pressure (9,6). The reason for the decrease is thought to be due to the fact that compression of the rock makes it harder for the bit to break it up. In order to keep the penetration rate at a reasonable level the pressure, and

hence the mud weight, should be kept as low as possible while allowing for the highest formation pressures to be encountered.

2. Solids content:

   Eckel (8) and Moore (20) have shown that the larger the solids content the slower the penetration rate. The reason for this appears to be because the particles interfere with the removal of the cuttings although the actual mechanism is unclear.

3. Mud viscosity:

   Increased drilling mud viscosity results in a slower rate of penetration (20). In the case of normal drilling muds the solids content and the viscosity are strongly interdependent and hence it might be thought that the viscosity was not an independent parameter. Moore (20), however, showed that the effect could be achieved using high viscosity glycerine. Moore suggested that the high viscosity resulted in lower fluid velocity in the vicinity of the cuttings at the bottom of the hole and hence the cuttings were not removed efficiently.

4. Oil content:

   Shales, especially, drill much more quickly when oil is added to the drilling mud (20). This effect probably results from the oil preventing "balling-up" of the bit. The economic advantage of adding oil is not always clear. The loss rate must be strictly controlled or else the cost of the oil used may

well be higher than the cost saving it produces through increased penetration rate.

The necessity of adequate hydraulic horsepower has long been recognized (1,14,20,25,28), but quantitative treatments are limited and generally hydraulic programs have been designed for a given rig to give maximum bit horsepower or impact force with the available pump input horsepower at maximum surface pressure. The thought is that, since actual requirements are unknown, the more horsepower the better (10). The function of the drilling fluid is to remove the cuttings from the area of the bit in order that fresh material may be drilled. Because of this it follows that the level of hydraulic horsepower required is related to penetration rate (28,3) and hole diameter (3) in that the volume of cuttings generated in unit time is proportional to the drilling rate and to the square of the hole diameter.

Circulation rate must be considered as the upward fluid velocity has to exceed the cuttings' slip velocity in order that the cuttings be removed from the hole (14).

Hopkin (14), discusses the factors affecting removal of cuttings during drilling,while the choice of mud properties and the hydraulic horsepower required,as dictated by differing hole conditions,is well documented by Reid (25).

The bit selection is, naturally, very important and depends mainly on the degree of hardness of the formation being drilled. The Hughes Tool Company (15) has discussed, in general terms, the bit re-

quirements for drilling different formation types. It is not, however, proposed in this work to discuss the intricacies of bit selection, as to this day it is a matter for a certain amount of trial and error.

The effects of weight and speed have been extensively investigated but it would appear from the variety of expressions that have been suggested that the processes involved are complicated and require careful analysis. Bielstein and Cannon (1) were among the early investigators and they noted that drilling rate appeared to vary approximately linearly with weight and somewhat less than linearly with rotary speed.

Similar results were obtained from the analysis of field results conducted for the American Association of Oil Well Drilling Contractors (30,7).

Moore (20) was the first to suggest an analytical method for finding the optimum weight and speed. He suggested that the drilling rate equation was:

$$\frac{dF}{dT} = k\ WN \qquad \text{(II-1)}$$

where, F = footage drilled
T = rotating time, hours
k = proportionality constant
W = weight on bit, 1000 lbs
N = rotary speed, RPM

Bit life limited by tooth wear was considered but no formula

was given so that optimization under this limiting condition was not considered. An equation for bearing life was proposed and so optimization was possible for a bearing controlled situation. The equation Moore suggested is

$$B = k^{'}/NW^{q} \qquad \text{(II-2)}$$

where, B = bearing life, hours

$k^{'}$ = proportionality constant

q = exponent on weight, Moore suggests 2.

King (16) subsequently confirmed the form of equation (II-2).

From previous data Moore found the values of the constants, and combining equations (II-1) and (II-2) with such factors as the rig cost, bit cost and trip time he was able to predict the optimum weight and speed at any point. The optimization technique suggested by Moore has been used extensively by other investigators on the more sophisticated equations developed later.

Much effort has been concentrated on the drilling rate equation. All the early investigations gave results of the form:

$$\frac{dF}{dT} = k\ f(W,N) \qquad \text{(II-3)}$$

Maurer (17,18) suggested that

$$f(W,N) = N(W - W_{o})^{2} \qquad \text{(II-4)}$$

for "perfect cleaning", by which he meant instantaneous removal of all the cuttings. The equation he proposed was:

$$\frac{dF}{dT} = \frac{N(W - W_o)^2}{H^2 \ S_1^{\ 2}} \qquad \text{(II-5)}$$

where, W_o = threshold weight before the bit penetrates the rock

H = hole diameter, inches

S_1 = formation drillability strength, constant.

In this way Maurer related penetration rate to weight per inch of bit diameter and made the constant S_1 independent of bit size.

Perhaps the most complicated theoretical analysis done to date is by Outmans(24). The equation for drilling rate that Outmans derived described the rate in terms of weight, rotary speed and hydraulic horsepower at the bit. The resulting equation is complex and contains several unknown constants which would have to be estimated from previous experience in the same area. The simplifications that arise when weight or rotary speed is the only variable give a general insight into field results. If weight is the only variable,

$$\frac{dF}{dT} = A[(1 + BW)^{1/2} - 1]^2 \qquad \text{(II-6)}$$

In this equation A and B are constants which include hydraulic horsepower and rotary speed terms. Similarly for variable rotary speed,

$$\frac{dF}{dT} = \frac{A'}{N} [(1 + B'N)^{1/2} - 1]^2 \qquad \text{(II-7)}$$

where A' and B' include hydraulic horsepower and weight terms.

Examination of the asymptotes of equation (II-6) gives,

1. $BW << 1$ then $(1 + BW)^{1/2} \simeq 1 + \frac{1}{2} BW$ (II-8)
2. $BW >> 1$ then $(1 + BW)^{1/2} \simeq (BW)^{1/2}$ (II-9)

Hence, for case 1,

$$\frac{dF}{dT} \simeq A(\frac{1}{2} BW)^2 \qquad \text{(II-10)}$$

and for case 2,

$$\frac{dF}{dT} \simeq ABW \qquad \text{(II-11)}$$

Outmans showed that BW is much less than 1 for most rocks drilled by the oil industry so that the linearity observed in field results was a reasonable approximation.

The equivalent equations in the case of variable rotary speed are:

$$\frac{dF}{dT} \simeq \frac{1}{4} A' B'^2 N \qquad \text{(II-12)}$$

when $B'N << 1$, and

$$\frac{dF}{dT} \simeq A'B' \qquad \text{(II-13)}$$

when $B'N >> 1$.

Thus the response to rotary speed will be at best, linear, and at worst, nil. The field results which show

$$\frac{dF}{dT} \alpha N^q \qquad \text{(II-14)}$$

where $q \leq 1$

are thus simply an approximation over a small range with q constant. This explains the reason for the variation in the exponent found by different investigators.

One other important point brought out by Outmans' work is the danger in examining one variable at a time as appears to have been done by all experimentalists. It should be noted that

$$\frac{dF}{dT} = k\ f(W,N)$$

does not imply

$$\frac{dF}{dT} = k\ g(W)\ h(N) \qquad \text{(II-15)}$$

where g(W) and h(N) are functions of weight and rotary speed respectively. This independence of the effects of weight and rotary speed cannot be theoretically justified and therefore the equations suggested may be slightly suspect.

The final step towards a comprehensive model was made by members of the Hughes Tool Company. Cunningham (5) first noted that, even when a homogeneous formation was being drilled, the proportionality constant of the drilling rate equation appeared to vary with the age of the bit. Galle and Woods (11) introduced the concept of bit dullness. The equations they developed are of the form:

$$\frac{dF}{dT} = \frac{k_1 f_1(W) g_1(N)}{h_1(D)} \qquad \text{(II-16)}$$

where D = normalized tooth wear

0 if new

1 if completely worn out.

$$\frac{dD}{dT} = \frac{k_2 f_2(W) g_2(N)}{h_2(D)} \qquad \text{(II-17)}$$

$$\frac{dB_x}{dT} = k_3 f_3(W) g_3(N) \qquad \text{(II-18)}$$

B_x = normalized bearing wear

0 if new

1 if completely worn out.

The assumptions made by Galle and Woods will, of course apply to any optimization procedure which is based on their equations. These assumptions are as follows.

1. Drilling cost is the summation of bit cost, rotating cost, connection cost and hoisting cost and these costs can be expressed equivalently in terms of rig hours.
2. Diamond bits are excluded from the analysis.
3. Bit life is limited by bearing failure or tooth wear or drilling rate or a combustion of these factors.
4. Circulating hydraulics are adequate and do not limit drilling rate.
5. The drilling rate is a function of only bit weight, rotary speed and degree of tooth dullness; that is for the purposes of this investigation the effects of pressure, lithology, fluid properties, hydraulics and drill string dynamics are assumed to be unalterable at any particular depth.
6. Within the range of rotary speed specified there are no restrictions brought about by prime mover performance.

Later investigations by Edwards (10) in 1964 and an on-well computer application by Young (31) in 1969 modified the Galle and Woods equations slightly but the industry (19) appears to consider that Galle and Woods equations are the best empirical set available at this time.

It cannot be said that the models proposed thus far are highly sophisticated, but the development is sufficiently far advanced to justify the use of optimization techniques.

III. General Theory

In view of the fact that the Galle and Woods equations are the best currently available, the various procedures discussed will be formulated in terms of these equations. For the sake of clarity, the original presentation of these equations by Galle and Woods is set out in the following section.

A. Summary of the Galle and Woods Equations.

1. Drilling Rate Equation

$$\frac{dF}{dT} = \frac{C_f \overline{W}^k r(N)}{(a(D))^p} \qquad \text{(III-A-1)}$$

where F = distance drilled by bit, feet

T = rotating time, hours

C_f = formation drillability factor

$\overline{W}$ = equivalent 7.88" bit weight = 7.88W/H

W = bit weight, 1000 lb.

H = bit or hole diameter, inches

k = weight exponent

$r(N) = \exp(-100/N^2)\ N^{0.428} + 0.2N(1 - \exp(-100/N^2))$

in the case of hard formations, and

$\exp(-100/N^2)N^{0.750} + 0.5N(1 - \exp(-100/N^2))$

in the case of soft formations

N = rotary speed, RPM

$a(D) = 0.928125D^2 + 6.0D + 1$

D = normalized tooth wear

p = exponent on $a(D)$

1.0 for flat-crested wear, 0.5 for self-sharpening wear and 0.0 for "button" bits.

The form of the functions in the above equation is such that the drilling rate increases as higher weights and rotary speeds are applied, and decreases as the bit dulls. The drilling rate is also proportional to the formation drillability parameter C_f which includes the effects of bit type, hydraulics, drilling fluid and the formation. It should be noted that, if it is decided that the bit type or the hydraulics or the mud system should be changed, the parameter will change.

Formations of a very soft nature, for which the penetration rate is not a linear function of weight, are covered, to a certain extent, by the use of the exponent k. The value of 0.6 for k was found to be the best compromise in these cases (11).

2. Rate of Dulling Equation

$$\frac{dD}{dT} = \frac{i(N)}{A_f \; a(D) \; m(\overline{W})} \qquad \text{(III-A-2)}$$

where A_f = formation abrasiveness factor

$i(N) = N + 4.438 \times 10^{-5} N^3$

$m(\overline{W}) = 1359.1 - 714.19 \log_{10} \overline{W}$

The rate of dulling increases, as would be expected, as higher weights and rotary speeds are applied. The rate of dulling decreases as the dullness increases. This happens because the conical shape of the bit tooth results in a larger area being available as the tooth wears. As in the drilling rate equation, where C_f was affected by down-hole conditions, A_f in this equation includes the effect of bit type, hydraulics, drilling fluid and formation, and will be altered if any of these factors is adjusted.

3. Bearing Life Equation

$$B = \frac{S\ l(\overline{W})}{N} \qquad \text{(III-A-3)}$$

where B = bearing life, hours

S = drilling fluid factor

$l(\overline{W}) = 21340./(1 + 0.03\overline{W})^{3.23}$

The bearing life decreases as greater weight or speed is used. In this form the bearing life equation is not particularly useful, and it is found convenient to use the fraction of bearing life consumed.

$$B_x = \frac{T}{B} = \frac{T\ N}{S\ l(\overline{W})} \qquad \text{(III-A-4)}$$

In this equation B_x is the fraction of total life expended. Assuming constant weight and speed this equation can be differentiated to give the rate of wear:

$$\frac{dB_x}{dT} = \frac{N}{S\ 1(\overline{W})} \qquad \text{(III-A-5)}$$

Again, it should be noted that the parameter S is a function of the bit type, mud and hydraulics, and would change if the hole conditions or bit type were altered. In the case of the sealed-bearing type of bit the parameter for a particular bit type should, in theory, always be the same. In practice, however, some variation may be expected because the bit action, and hence the forces on the bearing, is affected by down-hole conditions.

B. Calculation of the Parameters in the Galle and Woods Equations.

Before any optimization procedure can be used with the Galle and Woods equations it is necessary that the values of the three parameters C_f, A_f and S be known as a function of depth. The only way that the parameters can be found is from previous wells drilled in the same area. From bit runs made in the previous wells the following information is required:

1. Rotary speeds used during bit run - N_j
2. Weights used during bit run - $\overline{W}_j$

3. Footage drilled during each interval in which the speed and weight remained constant - ΔF_j
4. Time that each of the intervals occupied - ΔT_j
5. Final teeth condition - D_f
6. Final bearing condition - B_{xf}
7. Hole size - H
8. Formation response to rotary speed - soft or hard
9. Formation response to weight - soft or hard
10. Wear type - self-sharpening or flat-crested or "button"

It is also necessary to rearrange the Galle and Woods equations in a form which permits solution for the parameters.

Evaluation of the formation abrasiveness factor A_f is considered first.

From equation (III-A-2)

$$A_f \int_0^{D_f} a(x)\,dx = \int_0^{T_f} \frac{i(N)}{m(\overline{W})}\,dT \qquad \text{(III-B-1)}$$

In the practical case the finite approximation to the integral on the right hand side of equation (III-B-1) is used. Hence solving for A_f

$$A_f \doteq \frac{1}{U(D_f)} \sum_{j=1}^{n} \frac{i(N_j)\Delta T_j}{m(\overline{W}_j)} \qquad \text{(III-B-2)}$$

where the weight and speed are constant over the interval ΔT_j and

$$\sum_{j=1}^{n} \Delta T_j = T_f \qquad \text{(III-B-3)}$$

T_f - total rotating time

and

$$\int_0^{D_f} a(x)\,dx = U(D_f) \qquad \text{(III-B-4)}$$

D_f - dullness at the end of the bit run.

It should be noted that equation (III-B-2) has little meaning when "button" bits are used and there is negligible tooth wear. When this is the case the value of A_f tends to infinity and the rate of dulling to zero. If this is the case, it is better to set the rate of dulling to zero and ignore any slight dulling that occurs.

The drilling fluid parameter S is calculated from the integrated form of equation (III-A-5):

$$\int_0^{B_{xf}} dB_x = \frac{1}{S}\int_0^{T_f} \frac{N}{1(\overline{W})}\,dT \qquad \text{(III-B-5)}$$

where B_{xf} is the bearing wear at the end of the bit run. Rearranging equation (III-B-5) to solve for S:

$$S = \frac{1}{B_{xf}} \sum_{j=1}^{n} \frac{N_j \Delta T_j}{1(\overline{W}_j)} \qquad \text{(III-B-6)}$$

The parameter C_f is calculated differently for the several wear types. In the case of "button" bits the procedure is as follows.

From equation (III-A-1) with D = 0,

$$\frac{dF}{dT} = C_f \, \overline{W}^k r(N) \qquad \text{(III-B-7)}$$

or

$$\frac{\Delta F_j}{\Delta T_j} = C_f \, \overline{W}_j^k \, r(N_j) \qquad \text{(III-B-8)}$$

in finite difference form. Solving for C_f,

$$C_f = \frac{1}{n} \sum_{j=1}^{n} \frac{\Delta F_j}{\Delta T_j \, \overline{W}_j^k r(N_j)} \qquad \text{(III-B-9)}$$

Self-sharpening and flat-crested wear may be considered together. Combining equations (III-A-1) and (III-A-2),

$$\frac{dF}{dT}\,\frac{dT}{dD} = \frac{C_f \, \overline{W}^k r(N) \; A_f \; a(D) \; m(\overline{W})}{(a(D))^p \, i(N)} \qquad \text{(III-B-10)}$$

$$\frac{dF}{dD} = \frac{C_f \, \overline{W}^k r(N) \; m(\overline{W}) \, (a(D))^{1-p} \; A_f}{i(N)} \qquad \text{(III-B-11)}$$

By exactly the same procedure as shown when finding A_f equation (III-B-11) may be integrated and solved for C_f.

The answers for self-sharpening and flat-crested wear are slightly different because of the different value given to p in each

case.

If the wear is self-sharpening, p = 0.5

$$C_f = \frac{1}{A_f\ V(D_f)} \sum_{j=1}^{n} \frac{i(N_j)\Delta F_j}{\overline{W}_j^k\ r(N_j)\ m(\overline{W}_j)} \qquad \text{(III-B-12)}$$

where

$$V(D_f) = \int_0^{D_f} (a(x))^{1/2}\ dx \qquad \text{(III-B-13)}$$

If the wear is flat-crested, p = 1.0

$$C_f = \frac{1}{A_f D_f} \sum_{j=1}^{n} \frac{i(N_j)\ \Delta F_j}{\overline{W}_j^k r(N_j)\ m(\overline{W}_j)} \qquad \text{(III-B-14)}$$

Thus all the equations necessary for calculating the parameters have been derived. For convenience the necessary equations for each wear type are summarized below.

1. Self-sharpening wear - A_f from equation (III-B-2), S from equation (III-B-6) and C_f from equation (III-B-12).
2. Flat-crested wear - A_f from equation (III-B-2), S from equation (III-B-6) and C_f from equation (III-B-14).
3. "Button" bits - S from equation (III-B-6) and C_f from equation (III-B-9).

The derivation of the equations from which the parameters can be found includes one important assumption, namely that the parameters are unchanged throughout the bit run. It is impossible to discover the

variation of the parameters with depth unless the bit is pulled at various times during a run and the wear measured. Obviously this would be much too costly and the only method that can be used is to examine the drilling rate profiles of the bit runs for any sudden changes in rate. Combining these rate changes with knowledge of the lithology, the values of the parameters as a function of depth can be estimated in order that a better approximation to the actual conditions can be found. Certain of the bit runs will, in fact, be made through one formation type only, and these are used for control on the values of the parameters. In this way the accuracy of the parameters will be improved as more and more wells are drilled in the same area.

C. The Galle and Woods Optimization Procedure.

There has already been developed what might be called "point" optimization. The solution of the governing equations has been described (11,12,2), but in each case the solution obtained is not truly optimal, as several important points have been ignored. However, as this solution is used as a stepping off point for the other methods to be discussed, it will be examined in some detail.

The object is to find the combination of weight and rotary speed which results in the lowest cost, the weight and speed remaining constant throughout the bit run.

There are three possibilities, any of which may control the answer.

1. Teeth control bit life.
2. Bearings control bit life.
3. Drilling rate controls bit life.

Each of these conditions must be investigated when deciding on the optimum weight and speed.

The information required before the optimization may be carried out is as follows:

1. The parameters found by the method described in part B.
2. The rig cost - c_r - dollars/hour.
3. The bit cost - c_b - dollars.
4. Trip rate - t_r - time to go one foot (both directions) during trip.
5. The change time - t_c - time required to change bit at surface plus the time required to change the number of drill collars.

Before the Galle and Woods equations can be used in any optimization procedure it is necessary that they be converted into a form amenable to solution. In the case of self-sharpening or flat-crested wear this is done by expressing each equation in terms of changes in dullness.

Hence by combining equation (III-A-1) and the inverse of equation (III-A-2) we have:

$$\frac{dF}{dD} = \frac{A_f \; C_f \; m(\overline{W}) \; \overline{W}^k \; r(N)(a(D))^{1-p}}{i(N)} \qquad \text{(III-C-1)}$$

Inverting equation (III-A-2) gives:

$$\frac{dT}{dD} = \frac{A_f\ a(D)\ m(\overline{W})}{i(N)} \qquad \text{(III-C-2)}$$

Combination of equation (III-A-5) with the inverse of equation (III-A-2) gives:

$$\frac{dB_x}{dD} = \frac{A_f\ N\ a(D)\ m(\overline{W})}{S\ l(\overline{W})\ i(N)} \qquad \text{(III-C-3)}$$

It is convenient to simplify the above equations by defining the following three functions:

$$f_1(\overline{W},N) = \frac{A_f\ m(\overline{W})}{i(N)} \qquad \text{(III-C-4)}$$

$$f_2(\overline{W},N) = f_1(\overline{W},N)\ C_f\overline{W}^k r(N) \qquad \text{(III-C-5)}$$

$$f_3(\overline{W},N) = \frac{f_1(\overline{W},N)\ N}{S\ l(\overline{W})} \qquad \text{(III-C-6)}$$

Substitution of these functions in equations (III-C-1) through (III-C-3) gives:

$$\frac{dT}{dD} = f_1(\overline{W},N)\ a(D) \qquad \text{(III-C-7)}$$

$$\frac{dF}{dD} = f_2(\overline{W},N)(a(D))^{1-p} \qquad \text{(III-C-8)}$$

$$\frac{dB_x}{dD} = f_3(\overline{W},N)\ a(D) \qquad \text{(III-C-9)}$$

For any section drilled at constant weight and speed these three equations may be integrated as follows:

$$\Delta T_s = f_1(\overline{W},N) \int_{D_1}^{D_2} a(x)\ dx \qquad \text{(III-C-10)}$$

$$\Delta F_s = f_2(\overline{W},N) \int_{D_1}^{D_2} (a(x))^{1-p}\ dx \qquad \text{(III-C-11)}$$

$$\Delta B_{xs} = f_3(\overline{W},N) \int_{D_1}^{D_2} a(x)\ dx \qquad \text{(III-C-12)}$$

In these equations ΔT_s is the rotating time, ΔF_s is the footage and ΔB_{xs} is the change in normalized bearing wear which occurs when the dullness changes from D_1 to D_2 and the weight $\overline{W}$ and speed N are applied to the bit.

As previously stated this derivation applies to self-sharpening or flat-crested wear only. For "button" bits the derivation is simpler because the rate of dulling equation is omitted.

The inversion of equation (III-A-5) yields:

$$\frac{dT}{dB_x} = \frac{S\ 1(\overline{W})}{N} \qquad \text{(III-C-13)}$$

Combination of equation (III-A-1) (with D = 0) and equation (III-C-13) gives:

$$\frac{dF}{dB_x} = \frac{S\ 1(\overline{W})\ C_f\overline{W}^k r(N)}{N} \tag{III-C-14}$$

By defining two functions equations (III-C-13) and (III-C-14) are much simplified. Hence:

$$f_1'(\overline{W},N) = \frac{S\ 1(\overline{W})}{N} \tag{III-C-15}$$

$$f_2'(\overline{W},N) = f_1'(\overline{W},N)\ C_f\overline{W}^k r(N) \tag{III-C-16}$$

Substitution of these functions into equations (III-C-13) and (III-C-14) and integration between the two values of normalized bearing wear (during which the weight and speed remain constant) gives:

$$\Delta T_s = f_1'(\overline{W},N)(B_2 - B_1) \tag{III-C-17}$$

$$\Delta F_s = f_2'(\overline{W},N)(B_2 - B_1) \tag{III-C-18}$$

The object of the derivation is to obtain a formula for the cost of the bit run. In view of the fact that the rig cost is usually described in dollars per hour it is convenient to express the cost in terms of rig hours. Thus the trip time, rotating time and change time

can be used directly as cost items. The bit cost is divided by the rig cost per hour to get its equivalent cost in rig hours. Remembering that the trip rate is defined as the time per return foot, the simplest way of calculating trip time is to multiply the trip rate by the average depth of the bit run. With these points in mind, the total cost per foot can be expressed as follows:

$$C_t = (t_r(e_s + \Delta F_s/2) + c_b/c_r + t_c + \Delta T_s)/\Delta F_s \qquad \text{(III-C-19)}$$

C_t = cost, rig hours per foot
t_r = trip rate, hours per return foot
e_s = depth at start of bit run
c_b = bit cost, dollars
c_r = rig cost, dollars per hour
t_c = change time

Some of the terms in the above equation are functions of $\overline{W}$ and N, while the rest are constant. Separating these two types:

$$C_t = (c_c + t_r\Delta F_s(\overline{W},N)/2 + \Delta T_s(\overline{W},N))/\Delta F_s(\overline{W},N) \qquad \text{(III-C-20)}$$

where

$$c_c = t_r e_s + c_b/c_r + t_c \qquad \text{(III-C-21)}$$

Galle and Woods, by utilizing the calculus of variations, solved equation (III-C-20) for the optimum weight and speed. The de-

tails are fully discussed in their paper (12). Because of the complexity of the derivatives it was necessary to solve the resulting equations numerically. In view of this, a direct search technique could equally well have been used.

The step-by-step procedure by which the optimum values of $\overline{W}$ and N are found will now be outlined.

1. Choose the search area of $\overline{W}$ and N. If, after the search is completed, the best weight and/or speed coincide with the lower limits of the search area, then these limits may be too high, and the search should be repeated with different limits.
2. Calculate $f_1(\overline{W},N)$, $f_2(\overline{W},N)$ and $f_3(\overline{W},N)$ if the wear is self-sharpening or flat-crested; otherwise, calculate $f_1'(\overline{W},N)$ and $f_2'(\overline{W},N)$ and then proceed to step 5.
3. Assume final dullness is 1.

   Calculate B_{xf} from equation (III-C-12).

   If $B_{xf} \leq 1$, D_f is indeed 1 - proceed to step 4.

   If $B_{xf} > 1$, $D_f < 1$ and must be found. From equation (III-C-12):

$$B_{xf} = 1 = f_3(\overline{W},N) \int_0^{D_f} a(x)\,dx \qquad \text{(III-C-22)}$$

$$f_3(\overline{W},N) \int_0^{D_f} a(x)\,dx - 1 = 0 \qquad \text{(III-C-23)}$$

This equation can be solved for D_f using the Newton-Raphson

technique (4):

$$D_{f_{j+1}} = D_{f_j} - \frac{\int_0^{D_{f_j}} a(x)\, dx - \frac{1}{f_3(\overline{W},N)}}{a(D_{f_j})} \qquad \text{(III-C-24)}$$

Equation (III-C-24) is repeated until the convergence criterion is satisfied.

4. Calculate ΔT_s and ΔF_s from equations (III-C-10) and (III-C-11) respectively; go to step 6.
5. Calculate ΔT_s and ΔF_s from equations (III-C-17) and (III-C-18) respectively.
6. Calculate the cost per foot from equation (III-C-20).
7. Using a search technique, repeat steps 2 through 6 until the best combination of $\overline{W}$ and N has been found; these values are called $\overline{W}^*$ and N^*.
8. If a "button" bit is being used, proceed to step 10. Otherwise, reduce D_f by 5% and using the values $\overline{W}^*$ and N^* found in step 7 to start with, recompute the lowest value of cost per foot for this new value of D_f.
9. If the cost found in step 8 is greater than that from step 7, go to step 10. Otherwise, proceed with search on D_f to find the best value and the associated cost and $\overline{W}^*$ and N^*.
10. The procedure is complete and the best operating parameters and the resulting time and footage have been found.

It should be noted that it is not necessary to examine the

rate controlled situation for a "button" bit because, with no tooth wear, the rate does not alter with time.

This method, although simple, suffers from several grave disadvantages. The procedure does not allow for variation in A_f, C_f and S during a bit run. No allowance is made for formation changes. A hypothetical case will show how this can lead to errors. Suppose a soft formation of interval 600 feet is being drilled, followed by a very hard formation. The "point" optimum procedure is being applied and gives 400 feet as the optimum footage for the first bit and 410 feet for the second. The second bit is thus pulled approximately half way through its suggested run. The question then arises as to whether one bit could have drilled the interval more cheaply. Alternatively, two bits, each drilling half the distance, might be best. The reason for this uncertainty is that the method is truly applicable only where the section being considered is very large compared with the distance drilled by each bit.

One modification that can be made rather easily to the basic Galle and Woods method is to consider the intervals in groups. This restricts errors to the end of the last interval in the group. The procedure is essentially the same as previously outlined except that after each optimum has been found, the associated footage is checked to see whether it extends further than the end of the interval. If it does, the dullness and bearing conditions at the end of the interval are calculated and these values used as the starting points for optimization of the rest of the bit run.

In this way, reaching the end of an interval need not imply changing a bit, unless of course, it is the last interval of the group.

D. Modified Galle and Woods Method.

This modification of the previous method is designed to overcome all the objections of the previous method with the exception of variation in C_f, A_f and S. The well is divided into intervals where A_f, C_f and S are constant and each interval is optimized separately. This method can therefore be called "interval" optimization.

The procedure is outlined as follows:

1. Select the interval and the constants.
2. Using $\overline{W}_{min}$ and N_{min} calculate $f_1(\overline{W},N)$, $f_2(\overline{W},N)$ and $f_3(\overline{W},N)$ or $f_1'(\overline{W},N)$ and $f_2'(\overline{W},N)$ depending on bit type. If a "button" bit is being used go to step 4.
3. Assume $D_f = 1$.

   Calculate B_{xf} from equation (III-C-22).

   If $B_{xf} \leq 1$, $D_f = 1$ as assumed - proceed to step 4.

   Otherwise calculate D_f from equation (III-C-24).
4. Calculate ΔF_s from equation (III-C-11) or (III-C-18) depending on bit type.
5. Calculate the minimum number of bits required:

$$M_b = \text{Integer part}(\text{Interval}/\Delta F_s) + 1 \qquad \text{(III-D-1)}$$

6. For this number of bits calculate the distance drilled by each.

$$\Delta F_s = \text{Interval length/Number of bits} \qquad \text{(III-D-2)}$$

7. Calculate $f_1(\overline{W},N)$, etc., as before. If a "button" bit is being used go to step 11.

8. From equation (III-C-11):

$$f_2(\overline{W},N)\int_0^{D_f}(a(x))^{1-p}\,dx - \Delta F_s = 0 \qquad \text{(III-D-3)}$$

and again using the Newton-Raphson root finding technique:

$$D_{f_{j+1}} = D_{f_j} - \frac{\int_0^{D_{f_j}}(a(x))^{1-p}\,dx - \frac{\Delta F_s}{f_2(\overline{W},N)}}{a(D_{f_j})^{1-p}} \qquad \text{(III-D-4)}$$

Equation (III-D-4) is repeated until the convergence criterion is satisfied.

If $D_f > 1$, the values of W and N are outside the range of possibilities; go to step 13.

9. Calculate B_{xf} from equation (III-C-22).

If $B_{xf} > 1$, the values of $\overline{W}$ and N are outside the range of possibilities, therefore go to step 13.

10. Calculate ΔT_s from equation (III-C-10); go to step 12.

11. From equation (III-C-18):

$$B_{xf} = \frac{\Delta F_s}{f_2(\overline{W},N)} \qquad \text{(III-D-5)}$$

If $B_{xf} > 1$, go to step 13.

Calculate ΔT_s from equation (III-C-17).

12. Calculate the total cost of the interval and hold the values of the cost and associated weight and speed if the cost is the lowest to date.

The total cost of drilling the interval is n times the cost of one bit drilling at the average depth. This is because the symmetry means that the bit is above the mid point of the interval for exactly half the time, and below the mid point exactly half the time. Thus each trip can be considered to have been made from the mid point of the interval. The cost, in rig hours, is thus:

$$C_n = n\ (\Delta T_s + c_b/c_r + t_c + t_r\ (e_s + e_e)/2) \qquad \text{(III-D-6)}$$

13. Continue the search for the best values of $\overline{W}$ and N, (if other grid points remain), by repeating steps 7 through 12.

14. If not on the first cycle, compare the best cost with that found for the previous number of bits. If the latest cost is not smaller, go to step 16.

15. Increase the number of bits by one and return to step 6.
16. The optimum number of bits is one less than the current number.

This procedure thus finds the lowest cost of drilling an interval on the assumption that it is best that each bit drills the same distance.

An examination of this assumption will now be made.

Suppose an interval y feet in length is to be drilled, then:

$$T_y = \sum_{i=1}^{n} f(z_i) \qquad \text{(III-D-7)}$$

and

$$\sum_{i=1}^{n} z_i = y \qquad \text{(III-D-8)}$$

where z_i is the distance drilled by the i^{th} bit and T_y is the total rotating time for the distance y. Differentiating to find the approximate change in total time for small changes in the distances drilled:

$$\Delta T_y \doteq \frac{\partial f(z_i)}{\partial z_i} \Delta z_i \qquad \text{(III-D-9)}$$

(Note that double use of the subscript i implies summation).

Now since the sum of the distances is constant, the algebraic sum of the changes in the distances must be zero. That is:

$$\sum_{i=1}^{n} \Delta z_i = 0 \qquad \text{(III-D-10)}$$

If the derivative is an increasing function of z_i:

$$\frac{\partial f(z_{mean} + \delta)}{\partial(z_{mean} + \delta)} > \frac{\partial f(z_{mean})}{\partial z_{mean}} > \frac{\partial f(z_{mean} - \delta)}{\partial(z_{mean} - \delta)}$$

and hence ΔT_y will be positive for all displacements of z_i from z_{mean}. In fact, the derivative will be an increasing function of the distance drilled in almost all cases. This can easily be seen by considering the following example. If a bit can drill 100 feet when weight W_1 and N_1 are applied, it will be able to drill 110 feet only if the weight and/or the speed are reduced. This reduction means that the time to drill the first 100 will be longer than before and to drill the full 110 feet will be considerably greater than 110% of the first time. Hence the variation with footage is greater than linear so that the derivative is an increasing function of the distance drilled. The best policy, therefore, as far as the rotating time is concerned, is to drill the same distance with each bit.

The total trip time for n bits in an interval can be found by considering the following. The first trip, at the top of the interval, is one way only, so that the trip time is $t_r \, e_s/2$. At the end of the first bit run the depth is $e_s + z_1$ and as the trip is made both ways the trip time is $t_r(e_s + z_1)$. Similarly at the end of the second run the depth is $e_s + z_1 + z_2$ and the trip time is $t_r(e_s + z_1 + z_2)$. The trip time at the end of the j^{th} bit run is $t_r(e_s + z_1 + z_2 + \ldots + z_j)$. The final trip is one way and takes $t_r \, e_e/2$. The total time spent tripping is thus:

$$T_{trips} = t_r\,(e_s/2 + (e_s + z_1) + \ldots + (e_s + z_1 + \ldots + z_{n-1}) + e_e/2) \qquad \text{(III-D-11)}$$

It is convenient to replace $e_s/2$ by $e_s - e_s/2$ and to complete the series by adding and subtracting e_e (which is equivalent to $e_s + z_1 + \ldots + z_n$). When this has been done the total trip time is written as follows:

$$T_{trips} = t_r(e_s - e_s/2 + (e_s + z_1) + \ldots + (e_s + z_1 + \ldots + z_n) + e_e/2 - e_e)$$

$$= t_r(e_s(n + 1) - (e_s + e_e)/2 + \sum_{i=1}^{n} ((n + 1 - i)z_i) \qquad \text{(III-D-12)}$$

It is now possible to find the total cost of drilling an interval y drilled with n bits.

$$C_n = T_y + T_{trips} + n\; c_b/c_r + n\; t_c \qquad \text{(III-D-13)}$$

In general the form of T_y is not known. If, for example, it can be considered as a quadratic function of the distance drilled, the

cost is:

$$C_n = \sum_{i=1}^{n} (a_0 + a_1 z_i + a_2 z_i^2) + t_r(e_s(n + 1) -$$

$$(e_s + e_e)/2 + \sum_{i=1}^{n} ((n + 1 - i)z_i) + n\ c_b/c_r + n\ t_c \qquad \text{(III-D-14)}$$

Applying the method of Lagrangian multipliers to equations (III-D-14) and (III-D-8), as shown in Appendix A, the lowest cost is found to occur when:

$$z_i = z_{mean} + \frac{t_r}{4a_2}(2i - n - 1) \qquad \text{(III-D-15)}$$

Thus, if the rotating time can be represented by a quadratic function, the lowest cost occurs when the distances drilled by successive bits increases by $t_r/2a_2$.

That it will always be true the later bits should drill further can be seen from the following argument. It has been shown that the rotating time is minimized when the distances drilled are equal. Hence the only way of lowering the cost by altering the distances drilled from the mean is by lowering the tripping time more than the drilling time is increased. The only way of lowering the tripping time is by reducing the distance to be tripped. This can be achieved by making the distances drilled by the bits greater as one proceeds down the hole.

Hence in general:

$$z_1 < z_2 < z_3 < \ldots < z_n \qquad \text{(III-D-16)}$$

In the general case, the problem of allocating a distance to be drilled to each bit can be solved by means of dynamic programming.

Using the terminology of Nemhauser (22), it is required to minimize

$$C_n = \sum_{i=1}^{n} r_i(x_i,d_i) \qquad \text{(III-D-17)}$$

where

$$\sum_{i=1}^{n} d_i = y \qquad \text{(III-D-18)}$$

and

$$x_{i-1} = t_i(x_i,d_i) \qquad i = 1, n \qquad \text{(III-D-19)}$$

In these equations $r_i(x_i,d_i)$ is the cost of commencing in state x_i and making decision d_i, and $t_i(x_i,d_i)$ is the transformation function which results in state x_{i-1} when decision d_i is applied to the system in state x_i.

In this case the transformation is simply:

$$x_{i-1} = x_i - d_i \qquad \text{(III-D-20)}$$

Defining $f_i(x_i)$ as the minimum cost of an i-stage process

starting in state x_i, there exist the following recursive relations:

$$f_1(x_1) = \min_{d_1=x_1} r_1(x_1,d_1) \qquad \text{(III-D-21)}$$

$$f_i(x_i) = \min_{0\leq d_i\leq x_i} (r_i(x_i,d_i) + f_{i-1}(x_i - d_i)) \qquad \text{(III-D-22)}$$

In practice there will be an upper and lower limit on the value of d_i which will result in limits on state x_i. At the i^{th} stage:

$$i\ d_{min} \leq x_i \leq \min \begin{bmatrix} i\ d_{max} \\ y \end{bmatrix} \qquad \text{(III-D-23)}$$

There will also be a minimum number of stages:

$$M_s = \text{Integer part}(y/d_{max}) + 1 \qquad \text{(III-D-24)}$$

Equation (III-D-16) imposes further limits on d_i. It becomes (allowing for the fact that the last stage is designated number 1):

$$d_1 > d_2 > d_3 > \ldots > d_n \qquad \text{(III-D-25)}$$

Since

$$x_1 = d_1 + d_2 + \ldots + d_i \qquad \text{(III-D-26)}$$

it follows that:

$$d_i < x_i/i \qquad \text{(III-D-27)}$$

Hence only values of d_i which satisfy the above inequality need be examined.

The optimum number of stages is that which produces the lowest value of $f_i(y)$.

The cost function is:

$$r_i(x_i,d_i) = T_{d_i} + c_b/c_r + t_c + t_r(e_e - x_i + d_i/2) \quad \text{(III-D-28)}$$

This is the minimum cost associated with commencing a bit run x_i feet from the bottom of the interval, and pulling the bit after drilling d_i feet. The rotating time T_{d_i} is assumed to be already optimized, in other words, the best combination of $\overline{W}$ and N to drill d_i feet has previously been found.

The procedure for solution is as follows:

1. Calculate $f_1(x_1)$ for x_1 over the range specified in (III-D-23); set i = 2.
2. Calculate $f_i(x_i) = r(x_i,d_i^*) + f_{i-1}(x_i - d_i^*)$ where d_i^* is the value of the decision i which gives the lowest cost and d_i^* is found by a search on d_i.
3. If i is greater than M_s, go to step 4; otherwise, increase i by 1 and return to step 2.
4. If $f_i(y) < f_{i-1}(y)$, increase i by 1 and return to step 2.
   If $f_i(y) \geq f_{i-1}(y)$, decrease i by 1 and go to step 5.
5. The optimum value for the number of bits is the current value of i; trace back through the stages to get the optimal policy.

There are thus three possibilities which have to be tested in connection with "interval" optimization. These are:

1. Each bit drills the same distance.
2. Each bit drills a certain distance more than the previous one. In other words the increase from bit to bit is constant.
3. The distance drilled by each bit is allocated by a dynamic programming technique.

E. "Multi-interval" Optimization.

The methods described thus far have foundered when the parameters in the three controlling equations have changed. In some situations this is not important because a change in the parameters signifies a change in the rock type which requires a change in bit type. If this is the case, the bit has to be pulled and replaced by another type.

In many cases, however, the same type of bit is to be used in the new formation and it is then open to question as to whether, just because the parameters have changed, the best point to change bits is indeed where the next formation starts.

The method about to be discussed involves the problem of deciding where the optimum point is for changing the bit. The analogy between this problem and, for example, catalyst replacement is obvious. Dynamic programming can be used to find the best policy. The discussion that follows, closely parallels that of Roberts (26) in his catalyst replacement problem.

Some definitions are necessary:

B_i = the state of the normalized bearing wear at stage i.

D_i = the state of the normalized tooth dullness at stage i.

$f_i(D_i, B_i)$ = the minimum cost of an i stage process starting in state i and following an optimal policy. The quantity i refers to the number of stages remaining. In this problem a stage is characterized by a certain footage drilled.

$g_i(D_i, B_i, \overline{W}_i, N_i)$ = the cost of stage i for a system originally in state D_i, B_i and where $\overline{W}_i$ and N_i are the weight and speed chosen for stage i.

$h_i(D_{i+1}, B_{i+1}, \overline{W}_{i+1}, N_{i+1})$ = the transformation function which gives state D_i, B_i at stage i as a result of starting in state D_{i+1}, B_{i+1} at stage i+1 and applying $\overline{W}_{i+1}$ and N_{i+1} during stage i+1.

R_i = replacement cost at stage i.

Because each stage is a certain footage drilled, it is obvious that, after replacement, the stage footage remains to be drilled. As $f_i(0,0)$ is the cost of drilling the remaining i stages optimally, commencing with a new bit in the hole ready to start drilling, $f_i(0,0) + R_i$ is the cost of replacing a worn bit with a new one at the beginning of stage i and drilling the remaining i stages optimally.

The recursive relations are:

$$f_1(0,0) = \min_{\overline{W}_1, N_1} (g_1(0,0,\overline{W}_1,N_1)) \qquad \text{(III-E-1)}$$

$$f_1(D_1,B_1) = \min_{\overline{W}_1,N_1} \left[\begin{array}{c} \min\limits_{\overline{W}_1,N_1} (g_1(D_1,B_1,\overline{W}_1,N_1)) \\ R_1 + f_1(0,0) \end{array} \right] \qquad \text{(III-E-2)}$$

$$f_i(0,0) = \min_{\overline{W}_i,N_i} \left[\begin{array}{c} g_i(0,0,\overline{W}_i,N_i) \\ + \\ f_{i-1}(h_{i-1}(0,0,\overline{W}_i,N_i)) \end{array} \right] \qquad \text{(III-E-3)}$$

$$f_i(D_i,B_i) = \min_{\overline{W}_i,N_i} \left[\begin{array}{c} \min\limits_{\overline{W}_i,N_i} \left[\begin{array}{c} g_i(D_i,B_i,\overline{W}_i,N_i) \\ + \\ f_{i-1}(h_{i-1}(D_i,B_i,\overline{W}_i,N_i)) \end{array} \right] \\ R_i + f_i(0,0) \end{array} \right] \qquad \text{(III-E-4)}$$

The equations are evaluated in order and then equations (III-E-3) and (III-E-4) are repeated for successive values of i up to n.

The recursive equations have now been established in general form, but, of course, it is necessary to construct the equations in terms of the Galle and Woods equations for actual solution of the problem.

The transformation equations which apply in the case of self-sharpening or flat-crested wear will now be derived. Recalling that the final stage is given the number 1, what is required are equations relating D_{i-1} to D_i and B_{i-1} to B_i.

Rearranging equation (III-C-11) gives:

$$\int_0^{D_2} (a(x))^{1-p}\, dx - \int_0^{D_1} (a(x))^{1-p}\, dx - \Delta F_s / f_2(\overline{W},N) = 0 \qquad \text{(III-E-5)}$$

defining

$$Q_1 = \int_0^{D_1} (a(x))^{1-p}\,dx + \Delta F_s/f_2(\overline{W},N) \qquad \text{(III-E-6)}$$

and solving for D_2 using the Newton-Raphson root-finding technique, the recursive relationship becomes:

$$D_{2_{j+1}} = D_{2_j} - \left[\int_0^{D_{2_j}} (a(x))^{1-p}\,dx - Q_1\right]/(a(D_{2_j}))^{1-p} \qquad \text{(III-E-7)}$$

Rearranging equation (III-C-12) in order to solve for B_2:

$$B_2 = f_3(\overline{W},N) \int_{D_1}^{D_2} a(x)\,dx + B_1 \qquad \text{(III-E-8)}$$

Equations (III-E-7) and (III-E-8) can be generalized by inserting D_{i-1} for D_2, D_i for D_1, B_{i-1} for B_2 and B_i for B_1. The equations then become:

$$(D_{i-1})_{j+1} = (D_{i-1})_j - \frac{\int_0^{(D_{i-1})_j} (a(x))^{1-p}\,dx - Q_i}{(\,a((D_{i-1})_j\,))^{1-p}} \qquad \text{(III-E-9)}$$

$$B_{i-1} = F_3(\overline{W}_i,N_i) \int_{D_i}^{D_{i-1}} a(x)\,dx + B_i \qquad \text{(III-E-10)}$$

In the case of a "button" bit there is only one state variable and hence only one transformation equation. Using the same procedure by

which equation (III-E-10) was calculated, equation (III-C-18) may be transformed to give:

$$B_{i-1} = \Delta F_i / f_2'(\overline{W}_i, N_i) + B_i \tag{III-E-11}$$

This completes the derivation of the transformation equations. It is now necessary to express the cost of drilling stage i in terms of the rotating time. The equations are different for self-sharpening or flat-crested wear, and for "button" bits.

In the case of self-sharpening or flat-crested wear the cost is obtained by generalizing equation (III-C-10) to give:

$$\begin{aligned} g_i(D_i, B_i, \overline{W}_i, N_i) &= \Delta T_i \\ &= f_1(\overline{W}_i, N_i) \int_{D_i}^{D_{i-1}} a(x)\, dx \end{aligned} \tag{III-E-12}$$

In the case of "button" bits the equation (III-C-17) is generalized to give:

$$\begin{aligned} g_i(0, B_i, \overline{W}_i, N_i) &= \Delta T_i \\ &= f_1'(\overline{W}_i, N_i)\,(B_{i-1} - B_i) \end{aligned} \tag{III-E-13}$$

The replacement cost is

$$R_i = c_b/c_r + t_c + t_r e_i \qquad \text{(III-E-14)}$$

where e_i is the depth at the start of stage i.

This completes the derivation of the required equations, and it is now possible to outline the method of solution.

1. Set up a grid of normalized dullness and bearing wear for each stage of size L_i by M_i such that:

$$D_i(1) = B_i(1) = 0 \qquad \text{(III-E-15)}$$

$$D_i(L_i) = B_i(M_i) = 1 \qquad \text{(III-E-16)}$$

2. Calculate $g_1(0,0,\overline{W}_1^*,N_1^*)$ where $\overline{W}_1^*$ and N_1^* are the best values of $\overline{W}$ and N as found by a search technique. Thus $f_1(0,0)$ has been found.
3. Calculate the total cost of changing the bit before commencing the last stage and then drilling the stage:

$$C_r = R_1 + f_1(0,0) \qquad \text{(III-E-17)}$$

4. Evaluate $f_1(D_1,B_1)$ for all other values of D_1 and B_1 between 1 and L_1-1, and 1 and M_1-1, respectively. Obviously, for edge points, the bit MUST be changed, and hence

$$f_1(D_1,1) = f_1(1,B_1) = C_r \qquad \text{(III-E-18)}$$

5. Using the recursive relationships of equations (III-E-3) and (III-E-4), calculate $f_i(D_i, B_i)$ for each grid point of each stage until the final point is reached.
6. Work back through the data to get the optimal policy.

Several points should be noted:

1. Only values of $\overline{W}$ and N which allow transfer from D_i, B_i to D_{i-1}, B_{i-1} without making D_{i-1} and/or B_{i-1} exceed 1 are admissible at any stage.
2. Minimum values are assigned to $\overline{W}$ and N in order that the search area is cut down. If the minimum values violate condition (1) the bit will have to be changed.
3. In the case of "button" bits there is no dulling of the teeth, and hence the state variable D_i does not apply. The state variable grid is thus variable in B_i only; i.e. it is a vector in B_i.
4. Close examination of the controlling equations for self-sharpening or flat-crested wear shows that the state variable B_i does not affect the optimum procedure except where the bearings wear out first. If the teeth control bit life, therefore, the state variable grid can be reduced to a vector of D_i values only.

This completes the theory behind the methods investigated. The results obtained from each method will be discussed in the next section.

IV. Discussion of Results

A. Introduction

The difficulty in obtaining good data has restricted the amount of testing possible. Only recently has it become the practice of the major oil companies in Canada to keep sufficiently good records to make possible application of any of the techniques developed. Where records have been kept, sufficient data is still classified as confidential thus making it impossible to compare the methods with a well that was drilled without the benefit of optimization methods. Mobil Oil Canada Limited were prepared, however, to release a set of data which consisted of their interpretation of the equation parameters as a function of depth for a complete well, plus the necessary cost information, and how the formations might be grouped together. This information proved to be quite sufficient for testing the comparative worth of the different methods suggested.

Before the Galle and Woods equations were used, it was necessary to calculate approximating functions for:

1. $1(\overline{W})$
2. $\int_0^D (a(x))^{0.5}\, dx$

The function $1(\overline{W})$ was given in tabulated form by Galle and Woods (12). It was found that a hyperbolic fit was best.

The polynomial approximation of degree four represented the integral very well. It was discovered later that an exact solution[1] existed but the polynomial was retained because of its greater simplicity. The final results, together with the appropriate computer programs, can be found in Appendices B and C respectively.

Apart from the papers of Galle and Woods (11,12), and Billington and Blenkarn (2), no original data was available for testing the program that calculates the parameters. However, the program was thoroughly checked by calculating the parameters from the results of the optimization programs and comparing with the parameters actually used in these programs. The listing of the program and some typical results are given in Appendix D.

The Mobil data used in the optimization is listed in Appendix E. The programs and results for "point" optimization, "interval" optimization and "multi-interval" optimization are given in Appendices F, G and H respectively. Each of the methods will be discussed separately and then a comparison of the methods will be made.

B. Point Optimization

The simple version does not have a separate program written for it but can be taken as a special case of the "multi-interval" modification with each of the intervals considered separately. The improvement using the "multi-interval" modification was significant. Certain

---

[1] Hodgman, C.D., ed., "Mathematical Tables", Chemical Rubber Publishing Company, Cleveland, Ohio, 1962.

sections were not changed by the modification but, comparing the sections where a change did occur, the times involved were 1161 rig hours and 1132 rig hours respectively. This represented an improvement of 2.5% as a result of the modification whilst there was no change in the amount of computing time. Each computer run on the IBM 360/67 took approximately 0.1 minutes.

Because this program took so little time to run, it was convenient to use it for testing the different search techniques suggested by Wilde (29) and Rosenbrock and Storey (27). It was found that the more complex methods suggested such as "Partan" and Rosenbrock and Storey's rotating co-ordinates method did not improve on the simple gradient method (with weighting factors). The weighting factors which gave the best overall results were 1 for the weight and 4 for the rotary speed.

This method is quite adequate for the situation where neither the intervals nor the constants are known to any degree of accuracy. If used with a little common sense, under these conditions it should give reasonable results.

C. "Interval" Optimization

This was the second method tried. The most interesting result was that the modifications of the method proved to be wholly ineffective. The results were 1303.44 rig hours for the basic method, 1303.10 rig hours for the direct search method and 1303.51 for the dynamic programming procedure. Close examination of the answers suggested that the only case

where any appreciable savings might be made is where the drilling rate is extremely fast and the trip rate is extremely slow. Needless to say, this is a very unlikely combination, and so for practical purposes it can be concluded that making the distances drilled by each bit the same is quite satisfactory.

The dynamic programming technique gave the worst result by a small margin. In general, the program correctly assigned greater footage to the later bits but occasionally this was not true. For example in the interval 5070' to 6770' the second bit drills further than the rest. The reason for this would appear to be that the large number of interpolations result in small errors which are enough to influence the final answer.

The computing time for the simple procedure was 0.16 minutes, while that for the more complex methods was about 2.3 minutes in each case. Clearly, this alone would require a reasonable improvement when using the more complex methods.

D. "Multi-interval" Optimization

The most complex method is the "multi-interval" technique and, as might be expected, the largest number of problems were encountered in perfecting this method. Originally it was assumed that the 2-D grid of the state variables was necessary at all times, but as explained in Section III-E, it is only necessary to have a 2-D grid when the bearing wear is the controlling factor and there is appreciable wear on the teeth.

As the time required for this method increases exponentially with the number of dimensions, the time saved by this observation was considerable. In the case of the Mobil well, the time to compute the optimum procedure was reduced to less than 5 minutes from 30 minutes, while the accuracy was improved by using a larger number of elements in the dullness vector.

Obviously the starting point for a gradient search should not influence the resulting minimum. It was found, however, that at certain times it did make a large difference and so it became obvious that the function being searched was not unimodal. The reason for this became clear when the tables of optimum cost were examined.

As shown in Figure 1 the previous cost function increases up to a certain point and then becomes constant. The change occurs when it becomes cheaper to replace the bit before commencing the stage. Obviously once the bit is replaced the cost is the same no matter what the condition of the old bit was. Adding the cost of the current stage to the previous cost gives the total cost to date. The cost of the current stage decreases as the weight and speed increase but, simultaneously, the amount of wear during the stage increases, resulting in a higher previous cost. The form of the two functions is such that the total cost function has two minima, one internal, and one at the maximum allowed values of weight and rotary speed. Which of these two minima is the lower depends on the two curves which make up the composite curve.

Thus it is necessary to find both minima, and this can be done by starting at the lower limits of $\overline{W}$ and N and finding the internal minimum,

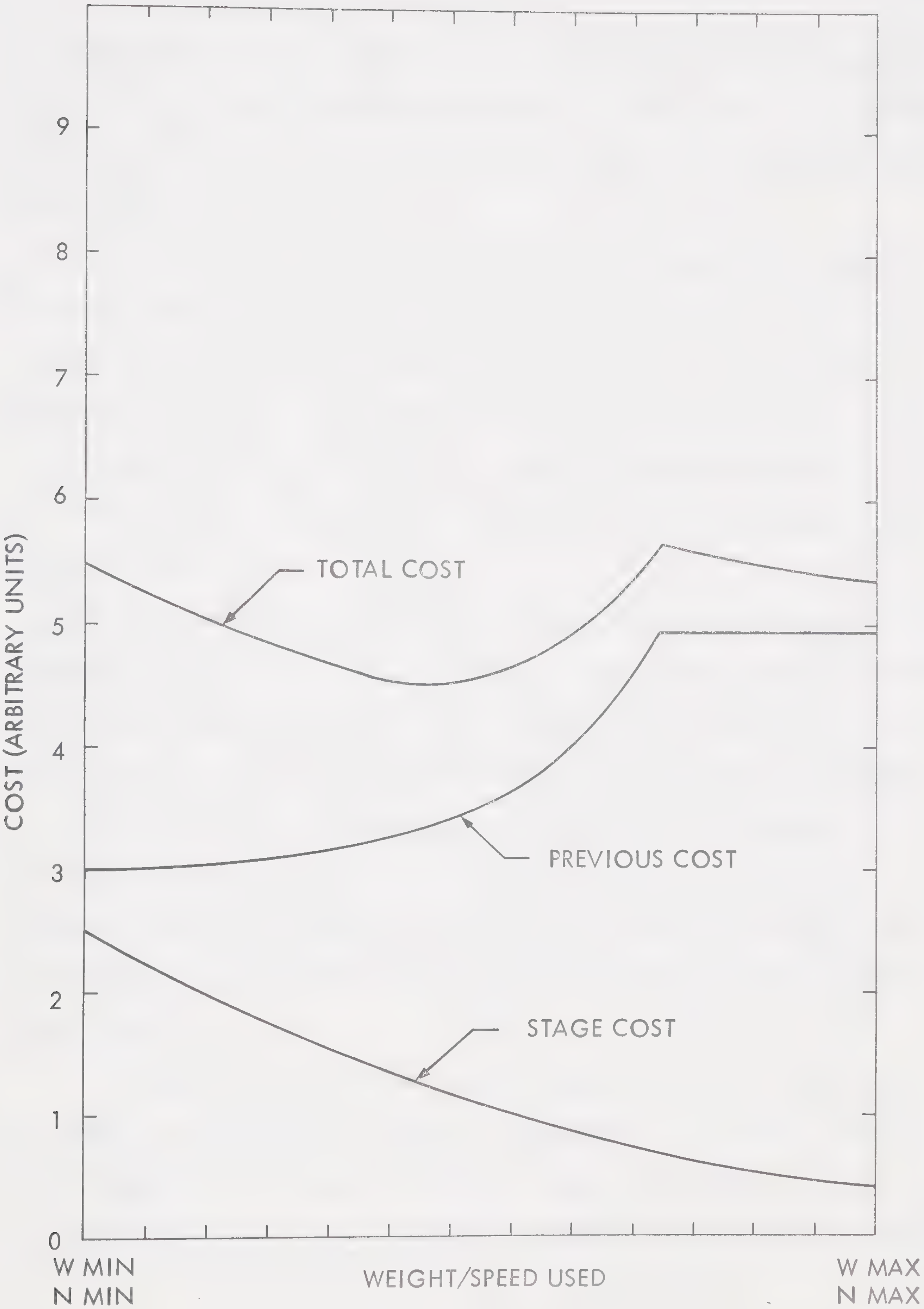

FIG. 1 COST VARIATION WITH WEIGHT AND SPEED

and starting at the upper limits of $\overline{W}$ and N and finding the minimum there ($\overline{W}_{max}$ and N_{max} may be in the forbidden region in which case it is necessary to find the edge of the allowed area). The lower of the two minima is then taken as the true optimum.

Because of the time factor involved in increasing the number of state variable grid points, extensive investigation was carried out into the effect of altering the number of grid points. In conjunction with this, variation in the number of intervals (or more correctly stages) and the degree of interpolation to be used in computing the previous cost were investigated. The results are presented in Figures 2 through 5 and Tables 1 through 6.

Figures 2 and 3 and Tables 1 through 3 show the effect of varying the number of intervals. The apparently haphazard plot is easily explained. The formation which was used for testing was of such a length that it was best drilled by 3 bits. Thus when the number of intervals is a multiple of 3, the cost is lower because each bit drills the same distance. If the number of intervals is not a multiple of 3, the bits cannot drill equal distances and the time taken is longer. This illustrates one point that must always be kept in mind, namely, that the number of stages must be chosen well.

Figures 4 and 5 and associated Tables 4 through 6 illustrate the effect of increasing the number of grid points. The general trend is for the results to improve as more grid points are used. It would also appear that the matrix size should be larger as the number of intervals

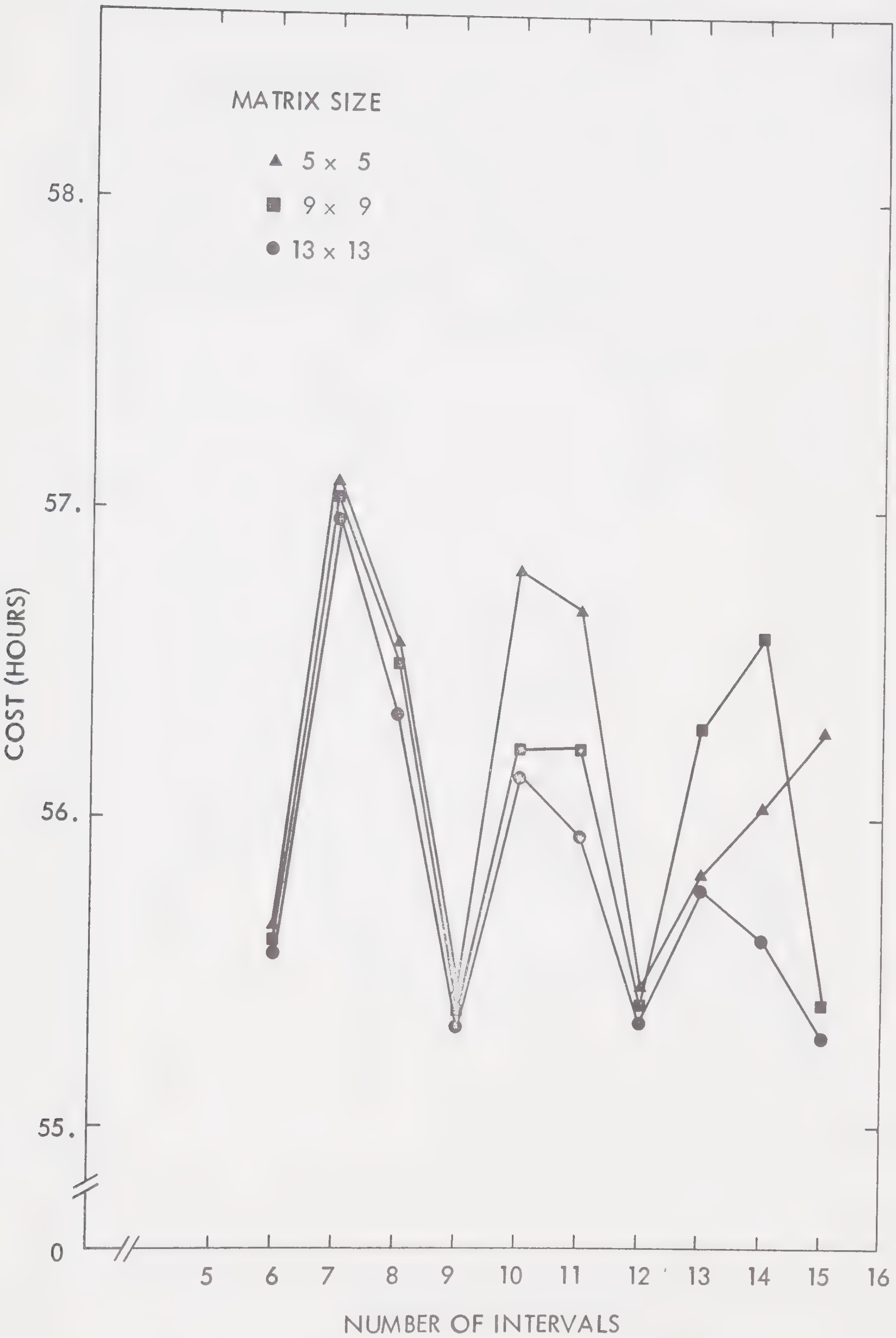

FIG. 2 COST VARIATION USING FIRST-ORDER INTERPOLATION

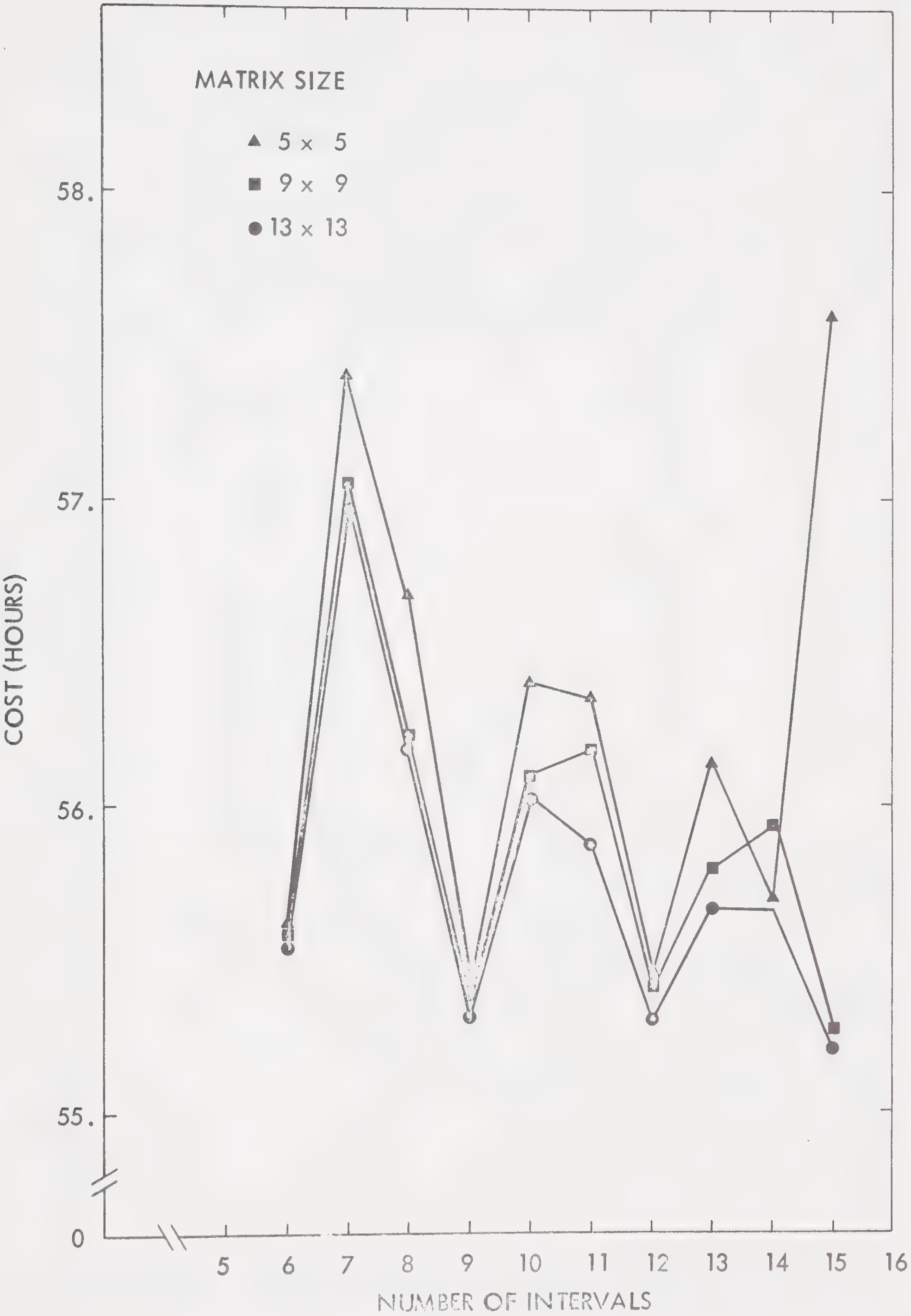

FIG. 3 COST VARIATION USING SECOND-ORDER INTERPOLATION

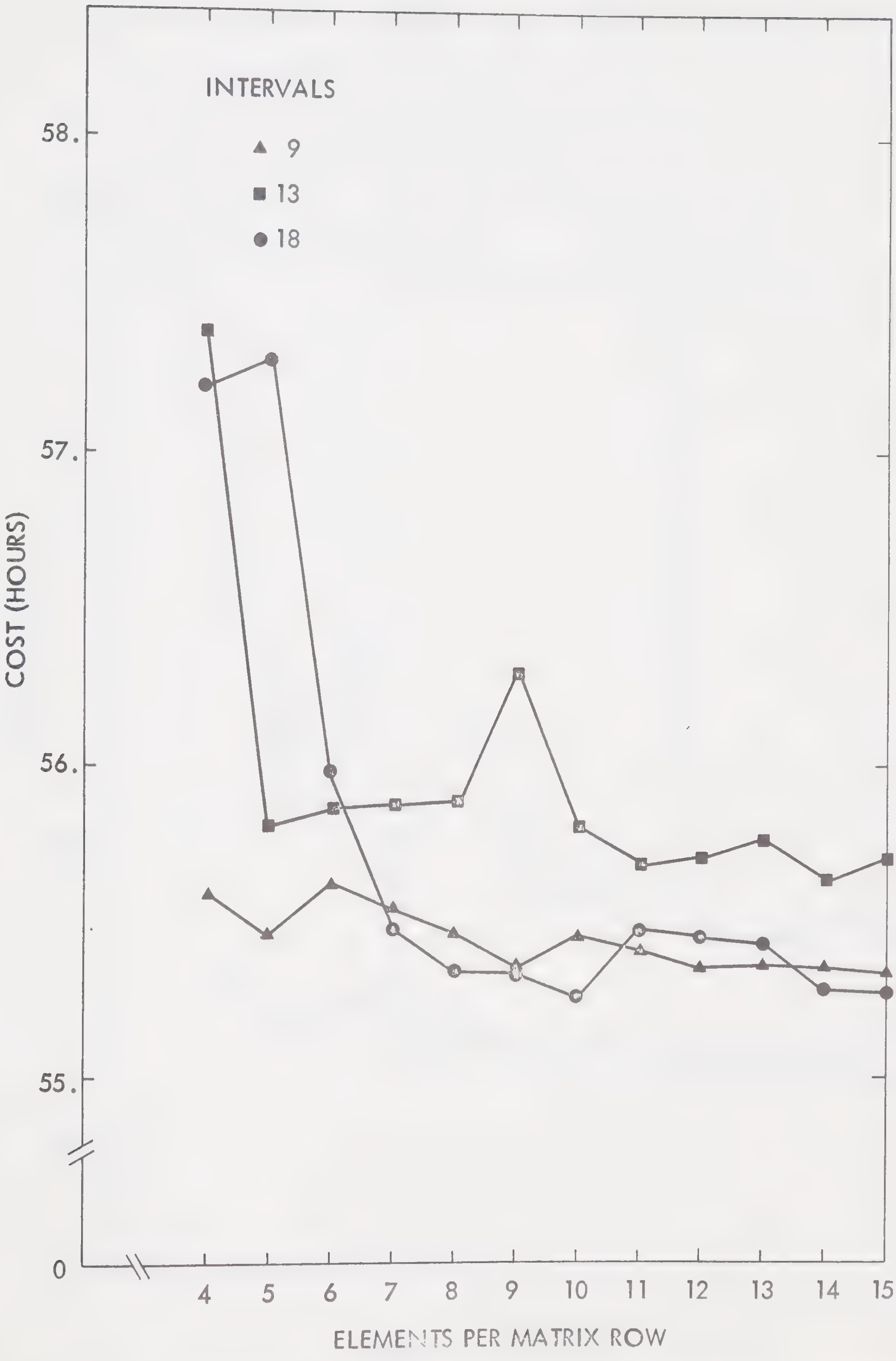

FIG. 4 EFFECT OF MATRIX SIZE AND FIRST-ORDER INTERPOLATION

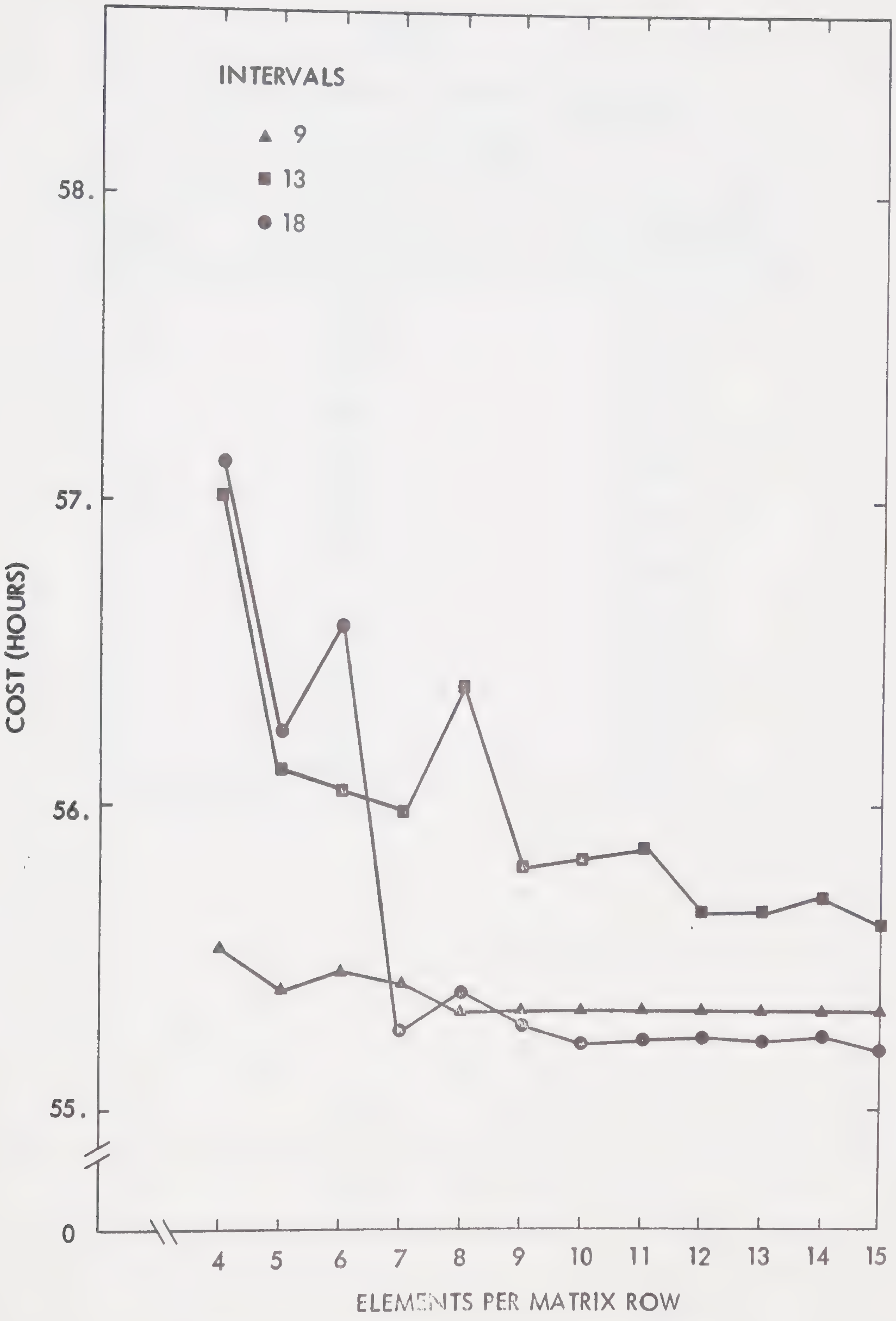

FIG. 5 EFFECT OF MATRIX SIZE AND SECOND-ORDER INTERPOLATION

Table 1

Cost Variation with Number of Intervals

Using a 5 x 5 Matrix

No. of Intervals	Cost Using 1st Order Interpolation	Cost Using 2nd Order Interpolation
6	55.64	55.63
7	57.11	57.44
8	56.58	56.69
9	55.46	55.40
10	56.79	56.64
11	56.64	56.38
12	55.53	55.44
13	55.80	56.16
14	56.04	55.64
15	56.36	57.51

Table 2

Cost Variation with Number of Intervals

Using a 9 x 9 Matrix

No. of Intervals	Cost Using 1st Order Interpolation	Cost Using 2nd Order Interpolation
6	55.60	55.57
7	57.08	57.06
8	56.53	56.26
9	55.37	55.35
10	56.25	56.11
11	56.25	56.20
12	55.50	55.41
13	56.31	55.80
14	56.59	55.93
15	55.38	55.26

Table 3

Cost Variation with Number of Intervals

Using a 13 x 13 Matrix

No. of Intervals	Cost Using 1st Order Interpolation	Cost Using 2nd Order Interpolation
6	55.58	55.56
7	56.96	56.99
8	56.30	56.24
9	55.38	55.35
10	56.12	56.05
11	55.91	55.85
12	55.46	55.28
13	55.77	55.63
14	55.61	55.67
15	55.27	55.23

Table 4

Cost Variation with Matrix Size Using 9 Intervals

Matrix Size	Cost Using 1st Order Interpolation	Cost Using 2nd Order Interpolation
4 x 4	55.58	55.57
5 x 5	55.46	55.40
6 x 6	55.63	55.47
7 x 7	55.55	55.43
8 x 8	55.46	55.35
9 x 9	55.37	55.35
10 x 10	55.48	55.36
11 x 11	55.43	55.35
12 x 12	55.35	55.35
13 x 13	55.38	55.35
14 x 14	55.38	55.36
15 x 15	55.36	55.35

Table 5

Cost Variation with Matrix Size Using 13 Intervals

Matrix Size	Cost Using 1st Order Interpolation	Cost Using 2nd Order Interpolation
4 x 4	57.36	57.03
5 x 5	55.80	56.16
6 x 6	55.88	56.06
7 x 7	55.90	55.98
8 x 8	55.89	56.41
9 x 9	56.31	55.80
10 x 10	55.79	55.83
11 x 11	55.67	55.88
12 x 12	55.70	55.65
13 x 13	55.77	55.63
14 x 14	55.64	55.72
15 x 15	55.69	55.61

Table 6

Cost Variation with Matrix Size Using 18 Intervals

Matrix Size	Cost Using 1st Order Interpolation	Cost Using 2nd Order Interpolation
4 x 4	57.21	57.17
5 x 5	57.32	56.29
6 x 6	55.97	56.62
7 x 7	55.47	55.27
8 x 8	55.33	55.38
9 x 9	55.36	55.30
10 x 10	55.26	55.22
11 x 11	55.49	55.24
12 x 12	55.44	55.27
13 x 13	55.45	55.25
14 x 14	55.28	55.26
15 x 15	55.29	55.20

per bit increases. For example, in Figure 5 an 8 x 8 matrix is required before stability occurs with 3 intervals per bit (9 overall), and a 10 x 10 matrix if 6 intervals per bit are used (18 overall). Also the results when using an odd number of intervals (i.e. in this case a non-multiple of 3) are more scattered.

The conclusions may be summarized as follows:

1. Quadratic interpolation may be considered slightly preferable to linear interpolation. The improvement is, however, small.
2. The size of the state variable matrix needed depends on the number of stages per bit. The suggested rule, which should have an adequate safety margin, is that the number of stages should be no greater than the number of row elements of the matrix divided by 2. Hence with a 9 x 9 matrix, the number of stages per bit should not exceed 4.
3. The number of stages chosen is important. If, for example, the optimum number of bits to drill an interval is three, then, the number of stages should be a multiple of three. This accounts for the strange curves obtained when plotting the cost against the number of stages. The effect of incorrectly choosing the number of stages produces a large error, but this error does decrease as the total number of stages increases.

The results of the optimization of the Mobil Oil data were quite encouraging, and it is believed that no difficulties should be encountered

in using the method. Although, as previously stated, the number of stages was important, it was found to be quite easy to select the number correctly, two or three trials being sufficient.

E. Comparison of the Methods

It can be seen from the results summarized in Table 7 that as the complexity of the method used increases, the amount of savings increases. The rig cost was \$200 per hour so that the saving between using the best and the worst method is approximately \$20,000. The charge for use of the IBM 360/67 at the University of Alberta is \$6 per minute which is low by commercial standards. However, even assuming a figure of \$10 per minute, the cost of a run using the most time-consuming method is \$50, while that for the simplest is about \$1. It can thus be seen that the saving that should be realized by use of the most complicated method is well justified.

The results show that the methods which have a practical use are numbers 2, 3 and 6 in Table 7. These are the modified "point" method, the "interval" method (in its simplest form) and the "multi-interval" method.

It is suggested that the modified "point" optimization procedure be used when there is little data available. It is very fast, and it would be possible to run the program several times, at little cost, using different intervals and parameters over the likely range in order to get an approximate optimum procedure. As more knowledge is gained about the

Table 7

Comparison of Different Optimization Techniques

Method No.	Optimization Type	Cost (Rig hours)	Saving over Method 1	Percentage Saving	Computing Time
1	"Point"	1366	-		0.10 min
2	Modified "point"	1337	29	2.1	0.10 min
3	"Interval"	1303	63	4.6	0.16 min
4	"Equal increment interval"	1303	63	4.6	2.35 min
5	"Interval by dynamic programming"	1304	62	4.5	2.30 min
6	"Multi-interval"	1265	101	7.3	4.88 min

geology, and more bit runs are made, the "interval" method becomes more appropriate, and finally, probably after three or four wells have been drilled, the "multi-interval" optimization should be used.

It is important to realize that, even if the predicted performance does not match the field performance very closely, this does not mean that the procedure is useless. It will always be true that the procedure can act as a guide towards better operating procedures.

One other use of the programs can be made. This is to examine

the effects of increasing the maximum weight and rotary speed. The programs can be run with very high maximum limits in order to see what the cost saving might be. This type of analysis is useful, for example, when hole deviation is a problem and the decision on the use of stabilizers is to be made. A good economic decision can be made by determining whether the cost of renting the stabilizers is recovered because of the faster drilling rates.

V. Conclusions and Recommendations

The object of this investigation was to examine the progress made to date in developing a satisfactory model of the drilling process and how optimization procedures could be used in conjunction with the model in order to reduce drilling costs.

The mathematical models available at this time have the disadvantage of including parameters which vary with operating conditions, so that changes in bit type or fluid properties alter the parameters. Thus any experimentation interferes with the optimization procedures.

However, assuming constant down-hole conditions are maintained from well to well, the following conclusions can be drawn:

1. Three useful methods have been developed for optimizing the drilling of a well. These are:
    - (i) "Point" optimization modified in order to consider groups of intervals.
    - (ii) "Interval" optimization with the bits drilling equal parts of the interval.
    - (iii) "Multi-interval" optimization.
2. The methods listed above are in order of increasing complexity and along with that complexity goes the need for better data. Hence the method used should be chosen according to the amount of data available.

3. The stage distribution for the "multi-interval" optimization has to be carefully selected. In order to do this, several trials will probably be necessary.
4. The cost of running the programs is small compared with the predicted saving.
5. The programs can be used to investigate the effects of raising the weight and speed limits. If a large cost saving results, this may spur investigation into how the limits might be raised.
6. Any new development should be incorporated in the optimization methods that have been derived.

It is suggested that the methods developed are as sophisticated as needed with the models developed thus far. It is therefore recommended that future work be concentrated on developing more adequate mathematical models of the drilling process.

The author believes that the methods developed, if used with care, can act as a guide toward good drilling procedures and that the cost saving should be considerable.

Nomenclature

A_f - formation abrasiveness factor.

B - bearing life, hours.

B_i - vector of normalized bearing wear values at stage i in a process or an element thereof.

B_x - fraction of bearing life expended.

B_{xf} - final fraction of bearing life expended.

ΔB_{xs} - change in normalized bearing wear over a section.

C_f - formation drillability factor.

C_n - total cost, expressed in rig hours, of drilling an interval with n bits.

C_r - total cost of replacing bit and drilling one stage .

C_t - total cost, expressed in rig hours per foot, of a bit run.

D - normalized tooth wear.

D_f - final normalized tooth wear.

D_i - vector of normalized tooth wear values at stage i of a process, or an element thereof.

F - distance drilled by bit, feet.

F_f - total distance drilled by bit, feet.

ΔF_i - length of stage i of a process.

ΔF_j - value of the j^{th} footage during a bit run.

ΔF_s - length of a section.

H - bit or hole diameter, inches.
L_i - number of elements in vector D_i.
M_b - minimum number of bits to drill an interval.
M_i - number of elements in vector B_i.
N - rotary speed, r.p.m.
N_i - vector of possible rotary speeds at stage i of a process, or an element thereof.
N_i^* - matrix of best rotary speeds at stage i of a process, or an element thereof.
N_j - value of the j^{th} rotary speed during a bit run.
R_i - cost of replacing bit at stage i.
S - drilling fluid factor.
T - rotating time, hours.
T_f - total rotating time, hours.
ΔT_i - rotating time for stage i.
ΔT_j - rotating time during the j^{th} interval of a bit run.
ΔT_s - rotating time to drill a section.
$U(D)$ - $0.309375\ D^3 + 3\ D^2 + D$.
$V(D)$ - $0.113328\ D^4 - 0.412768\ D^3 + 1.29230\ D^2 + 1.01978\ D$.
W - bit weight, 1000 lb.
$\overline{W}$ - equivalent 7.88" bit weight = $7.88\ W/H$.
$\overline{W}_i$ - vector of possible weights at stage i of a process, or an element thereof.
$\overline{W}_i^*$ - matrix of best weights at stage i of a process, or an element thereof.

$\overline{W}_j$ - value of the j^{th} weight during a bit run.

$a(D)$ - $0.928125D^2 + 6\ D + 1$.

c_b - bit cost, dollars.

c_c - fixed cost of a bit run, expressed in rig hours.

c_r - rig cost, dollars per hour.

d_i - vector of possible footages drilled at stage i, or an element thereof.

d_i^* - vector of best footages drilled at stage i, or an element thereof.

e - depth.

e_i - depth at beginning of stage i of a process.

e_e - depth of the bottom of an interval.

e_s - depth at the top of an interval.

$i(N)$ - $N + 4.438 \times 10^{-5}\ N^3$.

k - weight exponent in drilling rate equation.

- 1 in most cases but 0.6 for very soft formations.

$l(\overline{W})$ - $21340./(1 + 0.03\overline{W})^{3.23}$

$m(\overline{W})$ - $1359.1 - 714.19 \log_{10}\overline{W}$.

n - number of stages in a process.

p - exponent of $a(D)$ in drilling rate equation.

- 1.0 for flat-crested wear, 0.5 for self-sharpening wear and 0.0 for "button" bits.

$r(N)$ - $\exp(-100/N^2)\ N^{0.428} + 0.2\ N(1 - \exp(-100/N^2))$ in the case of hard formations, and $\exp(-100/N^2)\ N^{0.750} + 0.5\ N(1 - \exp(-100/N^2))$ in the case of soft formations.

t_c - time to change bit and alter the number of drill collars, hours.

t_r - trip rate, time to go one foot (both ways) during trip.

x_i - vector of footages remaining to be drilled at stage i of a process, or an element thereof.

y - length of an interval, feet.

z_i - distance drilled by i^{th} bit of a series, feet.

Bibliography

1. Bielstein, W.J., and Cannon, G.E., "Factors Affecting the Rate of Penetration of Rock Bits", A.P.I. Drilling & Production Practice, 17, 61, (1950).
2. Billington, S.A., and Blenkarn, K.A., "Constant Rotary Speed and Variable Weight for Reducing Drilling Cost", A.P.I. Drilling & Production Practice, 29, 52, (1962).
3. Bobo, R.A., "Simplified Rig-hydraulic Analysis", Oil & Gas J., 61, No. 23, 160, (1963).
4. Conté, S.D., "Elementary Numerical Analysis", McGraw-Hill, New York, 1965.
5. Cunningham, R.A., "Laboratory Study of the Effect of Rotary Speed on Rock-bit Performance", Oil & Gas J., 58, No. 28, 91, (1960).
6. Cunningham, R.A., and Eenink, J.G., "Laboratory Study of the Effect of Overburden, Formation and Mud Column Pressures on Drilling Rate of Permeable Formations", Trans. A.I.M.E., 216, 9, (1959).
7. Eckel, J.E., and Rowley, D.A., "The Effect of Rotary Speed on Penetration Rate", Oil & Gas J., 55, No. 47, 86, (1957).
8. Eckel, J.R., "The Effect of Mud Properties on Drilling Rate", A.P.I. Drilling and Production Practice, 21, 119, (1954).
9. Eckel, J.R., "Effect of Pressure on Rock Drillability", Trans. A.I.M.E., 213, 1, (1958).

10. Edwards, J.H., "Engineering Design of Drilling Operations", A.P.I. Drilling and Production Practice, 31, 39, (1964).

11. Galle, E.M., and Woods, H.B., "How to Calculate Bit Weight and Rotary Speed for Lowest Cost Drilling", Oil & Gas J., 58, No. 46, 167, (1960).

12. Galle, E.M., and Woods, H.B., "Best Constant Weight and Rotary Speed for Rock Bits", A.P.I. Drilling and Production Practice, 30, 48, (1963).

13. Garnier, A.J., and Van Lingen, N.H., "Phenomena Affecting Drilling Rate at Depth", Trans. A.I.M.E., 216, 232, (1959).

14. Hopkin, E.A., "Factors Affecting Cuttings Removal During Rotary Drilling", J. Pet. Tech., 19, 807, (1967).

15. Hughes Tool Company, Publication 10M, 1966.

16. King, G.R., "Effect of Fluid Environment on Rick-bit Bearing Performance", Oil & Gas J., 57, No. 47, 166, (1959).

17. Maurer, W.C., "The 'Perfect-Cleaning' Theory of Rotary Drilling", Trans. A.I.M.E., 225, 1270, (1962).

18. Maurer, W.C., "Bit-tooth Penetration under Simulated Bore-hole Conditions", J. Pet. Tech., 17, 1433, (1965).

19. Mobil Oil Canada Ltd., Personal Communication.

20. Moore, P.L., "Five Factors that Affect Drilling Rate", Oil & Gas J., 56, No. 40, 142, (1958).

21. Murray, A.S., and Cunningham, R.A., "The Effect of Mud Column Pressure on Drilling Rates", Trans. A.I.M.E., 204, 196, (1955).

22. Nemhauser, G.L., "Introduction to Dynamic Programming", Chapter III, 48, J. Wiley & Sons, Inc., New York, 1966.

23. , ibid, 55.

24. Outmans, H.D., "The Effect of Some Drilling Variables on the Instantaneous Rate of Penetration", Trans. A.I.M.E., 219, 137, (1960).

25. Reid, C.A., "Maximum Drilling Rate with Good Hole Conditions", A.P.I. Drilling and Production Practice, 33, 109, (1966).

26. Roberts, S.M., "Dynamic Programming in Chemical Engineering and Process Control", Chapter 2, Academic Press, New York, 1964.

27. Rosenbrock, H.H., and Storey, C., "Computational Techniques for Chemical Engineers", Pergamon, London, 1966.

28. Speer, J.W., "A Method for Determining Optimum Drilling Techniques", A.P.I. Drilling and Production Practice, 25, 130, (1958).

29. Wilde, D.J., "Optimum Seeking Methods", Prentice Hall, Englewood Cliffs, N.J., 1964.

30. Woods, H.B., and Galle, E.M., "The Effect of Weight on Penetration Rate", Oil & Gas J., 55, No. 47, 88, (1957).

31. Young, F.S., "Computarized Drilling Control", Trans. A.I.M.E., 246, 483, (1969).

Appendix A

Derivation of Footage Distribution

Appendix A

Derivation of Footage Distribution

From equation (III-D-14)

$$C_n = \sum_{i=1}^{n} (a_0+a_1z_i+a_2z_i^2) + t_r(e_s(n+1) - (e_s + e_e)/2)$$

$$+ t_r \sum_{i=1}^{n} (n+1-i)z_i + n\ c_b/c_r + n\ t_c \qquad (1)$$

For a given value of n, certain of the above terms do not vary. Hence defining:

$$C = t_r(e_s(n+1) - (e_s+e_e)/2) + n\ c_b/c_r + n\ t_c \qquad (2)$$

$$C_n = \sum_{i=1}^{n} (a_0+a_1z_i+a_2z_i^2) + t_r \sum_{i=1}^{n} (n+1-i)z_i + C \qquad (3)$$

It is desired to minimize C_n with respect to z_i, subject to the condition:

$$\sum z_i = y = \text{constant} \qquad (4)$$

Using the method of Lagrangian Multipliers (23) a new cost function is constructed:

$$C_n = \sum_{i=1}^{n} (a_0+a_1z_i+a_2z_i^2) + t_r \sum_{i=1}^{n} (n+1-i)z_i + C$$

$$+ L(\sum_{i=1}^{n} z_i - y) \qquad (5)$$

where L is the Lagrangian multiplier. Differentiating equation (5) with respect to z_i:

$$a_1 + 2a_2z_i + t_r(n+1-i) + L = 0 \qquad (6)$$

Adding the n equations obtained in this manner gives:

$$na_1 + 2a_2 \sum_{i=1}^{n} z_i + t_r \sum_{i=1}^{n} (n+1-i) + nL = 0 \qquad (7)$$

Now it is evident that

$$\sum_{i=1}^{n} (n+1-i) = n(n+1) - n(n+1)/2$$

$$= \frac{n}{2}(n+1) \qquad (8)$$

and $\sum_{i=1}^{n} z_i = y$ by definition.

Hence equation (7) may be modified to give:

$$a_1 + 2a_2y/n + t_r(n+1)/2 + L = 0 \qquad (9)$$

Hence solving for L gives:

$$L = - (a_1 + 2a_2y/n + t_r(n+1)/2) \tag{10}$$

Substituting in equation (6) and simplifying:

$$2a_2z_i - 2a_2y/n + t_r((n+1)/2-i) = 0 \tag{11}$$

Hence solving for z_i gives:

$$z_i = y/n + \frac{t_r}{4a_2}(2i - (n+1)) \tag{12}$$

$$z_i = z_{mean} + \frac{t_r}{4a_2}(2i-n-1) \tag{13}$$

Appendix B

Program and Results of $1(\overline{W})$ Fit

```
      C            MAIN

C          MAIN

C     THIS PROGRAM IS AN ADAPTATION OF AN I.B.M. APPLICATION
C     PROGRAM CALLED 1130 DECLINE CURVE ANALYSIS PROGRAM
C     (1130-MP-03X).  THE PROGRAM CAN BE USED TO APPROXIMATE
C     DATA BY EITHER AN EXPONENTIAL OR HYPERBOLIC CURVE.
C     THE ESSENTIAL DIFFERENCE BETWEEN THIS PROGRAM AND THE
C     I.B.M. PROGRAM  IS IN THE SEARCH TECHNIQUE.  THIS
C     PROGRAM USES THE FIBONACCI SEARCH METHOD WHILE THE
C     ORIGINAL USED AN INTERVAL HALVING PROCESS.  FOR DETAILS
C     OF THE METHOD THE MANUAL ISSUED BY I.B.M. CAN BE
C     CONSULTED.


      DIMENSION P(100),T(100),X(100),Y(100),V(100),QQ(100)
      DIMENSION IFIB(40)
    1 FORMAT(40I2)
    2 FORMAT(8F10.3)
  600 FORMAT(6(/)    ,21X,'BEST FIT CURVE FOR HYPERBOLIC'
     1,'[illegible]')
  601 FORMAT(6(/)    ,21X,'BEST FIT CURVE FOR EXPONENTIAL'
     1,'[illegible]')
  602 FORMAT (24X,'STANDARD DEVIATION IS ',F10.2,/)
  603 FORMAT(27X,'VALUE OF C1 IS ',F10.2,//,27X
     1,'VALUE OF C2 IS ',F10.2,/)
  604 FORMAT(27X,'VALUE OF C3 IS ',F10.2,/)
  605 FORMAT(17X,'X-VALUE         ACTUAL       COMPUTED       '
     1[illegible]
  606 FORMAT(17X,F6.1,6X,F7.0,5X,F7.0,6X,F7.0)
  607 FORMAT('1',//)
  608 FORMAT('1')

C     THE INPUT DATA IS AS FOLLOWS
C
C     NUMPR=NUMBER OF DATA POINTS
C     IOPT1=CHOICE OF CURVE
C      1=EXPONENTIAL
C      2=BOTH
C      3=HYPERBOLIC
C     IOPT2=DATA SPACING
C      0=EQUAL SPACING
C      1=UNEQUAL SPACING
C     IOPT3=NUMBER OF ITERATIONS
C      20 ITERATIONS ARE SUGGESTED.
C
C     IF THE DATA ARE EQUALLY SPACED ONLY THE LIMITS OF THE
C     X-VALUES NEED BE GIVEN.  OTHERWISE, ALL THE POINTS
C     MUST BE GIVEN.  THE Y-VALUES ARE, OF  COURSE, READ IN
C     IN BOTH CASES.
C
```

```
C                  MAIN           ...(CONT'D)
      READ(5,1)NUMBR,IOPT1,IOPT2,IOPT3
      IF(IOPT2.NE.0) GO TO 3
      READ(5,2) TIN,TOUT
      DELT=TOUT-TIN
      NUMBR1=NUMBR-1
      DELT=DELT/FLOAT(NUMBR-1)
      DO 4 I=1,NUMBR
      T(I)=DELT*FLOAT(I-1)+TIN
      GO TO 5
      READ(5,2) (T(I),I=1,NUMBR)
      READ(5,2) (P(I),I=1,NUMBR)
[illegible]
      FN=FLOAT(NUMBR)
      DO 6 I=1,NUMBR
      Y(I)=ALOG(P(I))
[illegible]
[illegible]
      REXP=0.
      ST=0.
      SST=0.
      SPT=0.
      DO 8 I=1,NUMBR
      ST=ST+T(I)
      SST=SST+T(I)*T(I)
    8 SPT=SPT+Y(I)*T(I)
[illegible]
[illegible]
[illegible]
      DO 9 I=1,NUMBR
      Q=P(I)-C1*EXP(-T(I)/C
    9 REXP=REXP+Q*Q
      GO TO (200,100),IOPT1
  100 A2=0.05
      J=1
      AA2=0.
      AA3=0.
      RR2=1.E+50
      RR3=1.E+50
[illegible]
  101 RHYP=0.
      ST=0.
      SST=0.
      SPT=0.
      DO 102 I=1,NUMBR
[illegible]
[illegible]
[illegible]
  102 [illegible]
      Q=FN*SST-ST*ST
      A1=(SST*SP-ST*SPT)/Q
```

```
      C                  MAIN              ...(CONT'D)

      A3=(FN*SPT-ST*SP)/Q
      DO 103 I=1,[illegible]
      Q=Y(I)-A1-A3*X(I)
  103 RHYP=RHYP+Q*Q
      GO TO (105,500,200),J
  105 RR1=RR2
      RR2=RR3
      RR3=RHYP
      AA1=AA2
      AA2=AA3
      AA3=A2
      IF(RR2-RR3)500,104,104
[illegible] A2=A2+A2
      GO TO 101
[illegible] IF(J.NE.1) GO TO 502
      IFIB(1)=1
      I[illegible]=1
      DO 501 I=3,IOPT3
  501 IFIB(I)=IFIB(I-1)+[illegible]
      FIB=FLOAT(IFIB(IOPT3))
      J=2
      JJ=1
  502 GO TO (590,591,592,593,594),JJ
  590 RANGE=AA3-AA1
      DEL=RANGE*FLOAT(IFIB(IOPT3-2))/FIB
      A2=AA1+DEL
      X1=A2
      JJ=2
      GO TO 101
[illegible] FX1=RHYP
      A2=AA3-DEL
      Y1=A2
      [illegible]
      GO TO 101
  592 [illegible]
      [illegible]=IOPT3-3
      KKK=IOPT3-2
      DO 520 II=1[illegible]
      [illegible]
      [illegible]
      IF(LL.EQ.1) DEL=DEL*0.9
      IF(FX1.GT.FY1) GO TO 523
      AA3=Y1
      Y1=X1
      FY1=FX1
      [illegible]
      A2=X1
      JJ=4
      GO TO 101
  593 FX1=RHYP
```

```
523 AA1=X1
    X1=Y1
    FX1=FY1
    [illegible]
    [illegible]
    [illegible]
    GO TO 101
594 FY1=RHYP
[illegible] CONTINUE
    IF(FX1.GT.FY1) GO TO 536
    A2=X1
    J=3
    GO TO 101
[illegible] A2=Y1
    J=3
    GO TO 101
[illegible] GO TO (2[illegible]
[illegible] IF(RHYP.LT.REXP) GO TO 203
[illegible] RR1=0.
    [illegible]
    [illegible]
    [illegible]
    [illegible]
    [illegible]
204 [illegible]
    [illegible]
    [illegible]
    [illegible]
    [illegible]
    [illegible]
    DO 205 I=1,NUMBR
    KOUNT=KOUNT+1
    IF(KOUNT.LE.57)  GO TO 206
    [illegible]
    WRITE(6,605)
    KOUNT=10
    KOUNT=10
[illegible] WRITE(6,606) T(I),P(I),V(I),[illegible](I)
[illegible] CONTINUE
    STOP
203 WRITE(6,600)
    WRITE(6,600)
    C1=EXP(A1)
    [illegible]
    C2=C3/A2
    RR1=0.
    DO 207 I=1,NUMBR
    V(I)=C1*(1.+C3*T(I)/C2)**((-1.)/C3)
    [illegible]
```

```
207 RR1=RR1+QQ(I)*QQ(I)
    [illegible]
    WRITE(6,602) RR1
    WRITE(6,603) C1,C2
    [illegible]
    WRITE(6,605)
    KOUNT=19
    DO 208 I=1,NUMBR
    KOUNT=KOUNT+1
    IF(KOUNT.LE.57)  GO TO 209
    [illegible]
    WRITE(6,605)
    KOUNT=10
209 WRITE(6,606) T(I),P(I),V(I),QQ(I)
208 CONTINUE
    STOP
    END
```

```
          BEST FIT CURVE FOR HYPERBOLIC DECLINE

              STANDARD DEVIATION IS        37.16

                  VALUE OF C1 IS    21336.87
                  VALUE OF C2 IS       10.20
                  VALUE OF C3 IS        0.31

   X-VALUE        ACTUAL       COMPUTED       DIFFERENCE
                  Y-VALUE      Y-VALUE

     15.0          6240.         6095.          145.
     16.0          5840.         5703.          137.
     17.0          5440.         5344.           96.
     18.0          5080.         5014.           66.
     19.0          4750.         4710.           40.
     20.0          4439.         4430.            9.
     21.0          4170.         4172.           -2.
 [illegible]    [illegible]   [illegible]    [illegible]
     23.0          3680.         3711.          -31.
     24.0          3470.         3506.          -36.
     25.0          3270.         3315.          -45.
     26.0          3080.         3138.          -58.
     27.0          2910.         2973.          -63.
     28.0          2770.         2819.          -49.
     29.0          2630.         2676.          -46.
     30.0          2496.         2542.          -46.
     31.0          2370.         2416.          -46.
     32.0          2260.         2299.          -39.
     33.0          2160.         2189.          -29.
     34.0          2060.         2086.          -26.
     35.0          1963.         1989.          -26.
     36.0          1880.         1898.          -18.
 [illegible]    [illegible]   [illegible]    [illegible]
 [illegible]    [illegible]   [illegible]    [illegible]
     39.0          1650.         1655.           -5.
     40.0          1578.         1584.           -6.
     41.0          1515.         1516.           -1.
     42.0          1460.         1452.            8.
     43.0          1400.         1392.            8.
     44.0          1340.         1335.            5.
     45.0          1288.         1281.            7.
     46.0          1240.         1229.           11.
     47.0          1195.         1181.           14.
     48.0          1150.         1135.           15.
     49.0          1105.         1091.           14.
     50.0          1063.         1049.           14.
     51.0          1025.         1010.           15.
     52.0           988.          972.           16.
```

X-VALUE	ACTUAL Y-VALUE	COMPUTED Y-VALUE
53.0	[illegible]	[illegible]
54.0	[illegible]	[illegible]
55.0	[illegible]	[illegible]
56.0	[illegible]	[illegible]
57.0	[illegible]	[illegible]
58.0	[illegible]	[illegible]
59.0	766.	[illegible]
60.0	[illegible]	[illegible]
61.0	714.	[illegible]
62.0	[illegible]	[illegible]
63.0	[illegible]	[illegible]
64.0	642.	[illegible]
65.0	[illegible]	[illegible]
66.0	[illegible]	[illegible]
67.0	[illegible]	[illegible]
68.0	[illegible]	[illegible]
69.0	[illegible]	[illegible]
70.0	[illegible]	[illegible]
71.0	[illegible]	[illegible]
72.0	[illegible]	[illegible]
73.0	[illegible]	[illegible]
74.0	[illegible]	[illegible]
75.0	[illegible]	[illegible]

Appendix C

Program and Results of Integral Fit

```
      C           MAIN

C           MAIN

C     THIS PROGRAM WAS WRITTEN IN ORDER TO FIND A POLYNOMIAL
C     APPROXIMATION FOR THE INTEGRAL OF THE SQUARE ROOT OF
C     DULLA(D) (EVALUATED BETWEEN D=0 AND D=D).  THE STANDARD
C     LEAST SQUARES TECHNIQUE IS USED AND THE FIT OF THE
C     FIT OF POLYNOMIALS FROM DEGREE 2 THROUGH 5 WAS TESTED.
C     THE SOLUTION OF THE SIMULTANEOUS EQUATIONS, FROM WHICH
C     THE CONSTANTS OF THE POLYNOMIALS ARE FOUND, WAS
C     ACHIEVED USING THE GAUSS-SEIDEL TECHNIQUE.
C
C
C     THERE IS NO DATA INPUT
C
C     INTERNAL VARIABLES
C
C     D1 - VECTOR OF D VALUES WITH WHICH THE VARIOUS VALUES
C          OF THE INTEGRAL WERE EVALUATED AND FITTED.
C     DIN1(J) - VALUE OF THE INTEGRAL CORRESPONDING TO D1(J)
C     D2 - VECTOR OF D VALUES WITH WHICH THE ACCURACY OF THE
C          FIT WAS TESTED.
C     DIN2(J) - VALUE OF THE INTEGRAL CORRESPONDING TO D2(J)
C

      DIMENSION D1(30),D2(30),DIN1(30),DIN2(30),A(10)
  600 FORMAT('1',6(/),31X,'ORDER OF FIT IS ',I2,/)
  601 FORMAT(27X,'VALUE OF A',I1,' IS ',F10.7,/)
  602 FORMAT(17X,'Y-VALUE         ACTUAL      COMPUTED      '
     1,'DIFFERENCE',/,29X,'Y-VALUE',5X,'Y-VALUE',/)
  [illegible] FORMAT (/,24X,'STANDARD DEVIATION IS ',F10.7,/)
  603 FORMAT(17X,F6.3,4X,F10.7,3X,F10.7,4X,F10.7)
  610 FORMAT('1')
      [illegible]
      [illegible]
      [illegible]
      DIN2(1)=0.
      DO 1 J=2,21
      [illegible]
      D2(J)=D2(J-1)+.1
      [illegible]
      [illegible]
    1 [illegible]
      DO 3 N=2,5
      CALL BEST(DIN1,D1, 9,N,A)
      [illegible]
      WRITE(6,600)
      DO 4 J=1,NMAX
    4 WRITE(6,601) J,A(J)
      SUM=0.
      WRITE(6,602)
```

```
      C                 MAI

    DO 5 J=1,21
    ANS=A(N+1)
    DO 6 K=1,N
    L=N+1-K
    ANS=ANS*D2(J)+A(L)
    [illegible]
    SUM=SUM+Y*Y
    WRITE(6,603) D2(J),DIN2(J),ANS,Y
    CONTINUE
    SUM=(SUM/21.)**.5
    WRITE(6,604) SUM
    WRITE(6,610)
    CONTINUE
    STOP
    END
```

FUNCTION SIM1

```
      FUNCTION SIM1(DULL)
C
C     THIS FUNCTION SUBPROGRAM INTEGRATES THE SQUARE ROOT OF
C     DULLA(D) BETWEEN 0 AND D USING SIMPSON'S RULE.
C
      DULLAH(D)=(1.0+(6.0+.928125*D)*D)**0.5
      [illegible]
      N2=2*IFIX(DULL/0.005)
      H=DULL/FLOAT(N2)
      NU=N2/2-1
      S=1.+4.*DULLAH(H)
      DO 3 I=1,NU
      S=2.*DULLAH(H*FLOAT(2*I))+4.*DULLAH(H*FLOAT(2*I+1))+S
      SIM1=(S+DULLAH(DULL))*H/3.
      GO TO 4
    1 SIM1=0.
    4 RETURN
      END
```

SUBROUTINE BEST

```
      SUBROUTINE BEST(YPOIN,POIN,J,N,A)
C     THIS SUBPROGRAM CALCULATES, FROM THE INPUT DATA, THE
C     ELEMENTS OF THE MATRIX AA AND THE COLUMN VECTOR C IN
C     ORDER THAT THE EQUATION (AA)(A)=C MAY BE SOLVED FOR
C     THE COLUMN VECTOR A.
C

      DIMENSION YPOIN(30),POIN(30),A(10),SUM(30),SUMY(30)
      DIMENSION C(10),AA(10,10),Y(30)
      M=2*N+1
      NMAX=N+1
      DO 40 L=1,M
      SUM(L)=0.
   40 SUMY(L)=0.
      DO 50 K=1,J
      Y(1)=1.
      DO 60 L=1,M
      SUM(L)=SUM(L)+Y(L)
      SUMY(L)=SUMY(L)+Y(L)*YPOIN(K)
      Y(L+1)=Y(L)*POIN(K)
   60 CONTINUE
   50 CONTINUE
      DO 70 K=1,NMAX
      DO 80KK=1,NMAX
      L=K+KK-1
   80 AA(K,KK)=SUM(L)
   70 CONTINUE
      DO 90 K=1,NMAX
   90 C(K)=SUMY(K)
      CALL GASE(A,AA,C,NMAX)
      RETURN
      END
```

SUBROUTINE GASE

```
      SUBROUTINE GASE(X,A,B,N)
C     THIS SUBPROGRAM CALCULATES THE SOLUTION TO N
C     SIMULTANEOUS EQUATIONS BY THE GAUSS-SEIDEL TECHNIQUE.
C
      DIMENSION A(10,10),B(10),X(10)
      DO 50 I=1,N
   50 X(I)=0.
   99 BIG=0.
      DO 100 I=2,N
  102 SUM=0.
      IF(I.EQ.1) GO TO 105
      LAST=I-1
      DO 106 J=1,LAST
  106 SUM=SUM+A(I,J)*X(J)
      IF(I.EQ.N) GO TO 103
  105 INITL=I+1
      DO 107 J=INITL,N
  107 SUM=SUM+A(I,J)*X(J)
  103 TEMP=(B(I)-SUM)/A(I,I)
      IF(ABS((TEMP-X(I))/TEMP).GT.BIG)
     1BIG=ABS((TEMP-X(I))/TEMP)
  100 X(I)=TEMP
      IF(BIG.LT.2.E-07) RETURN
      GO TO 99
      END
```

```
              ORDER OF FIT IS  4

          VALUE OF A1 IS   0.0

          VALUE OF A2 IS   1.0117752

          VALUE OF A3 IS   1.2[illegible]

          VALUE OF A4 IS  -[illegible]

          VALUE OF A5 IS   [illegible]

  X-VALUE          ACTUAL            COMPUTED          DIFFERENCE
                                     Y-VALUE

   0.              0.0               0.                0.
   0.100           0.1131902         0.1144094        -0.00[illegible]
   0.200           0.2712304         [illegible]      -0.00[illegible]3
   0.300           0.4127117         0.4122341         0.0004777
   0.400           0.5914192         0.5911103         0.000[illegible]
   [illegible]     0.7[illegible]    0.7[illegible]    0.00[illegible]
   [illegible]     1.0[illegible]    1.0[illegible]    [illegible]
   0.700           [illegible]       [illegible]      -0.00[illegible]
   [illegible]     1.477[illegible]  1.477[illegible] -0.00[illegible]
   0.900           1.72[illegible]   [illegible]       [illegible]
   [illegible]     [illegible]       2.0126410         0.0[illegible]
   [illegible]     2.2[illegible]    2.2[illegible]   -0.0[illegible]
   1.200           2.[illegible]     2.5[illegible]   -0.00[illegible]
   1.300           2.[illegible]     [illegible]      -0.[illegible]
   1.400           3.247141          3.2[illegible]   -0.0[illegible]
   1.500           3.[illegible]     3.6[illegible]   -0.0[illegible]
   1.600           3.[illegible]     3.[illegible]    -0.[illegible]
   [illegible]     4.30[illegible]   [illegible]      -[illegible]
   1.800           4.6[illegible]    [illegible]       [illegible]
   1.900           5.[illegible]     5.2[illegible]   -0.17[illegible]
   2.000           [illegible]       5.71[illegible]  -0.23[illegible]

        STANDARD DEVIATION IS  0.[illegible]
```

Appendix D

Program and Results of the Constants Calculations

```
            MAIN

C     THE FUNCTION OF THIS PROGRAM IS TO TAKE DATA FROM
C     A WELL PREVIOUSLY DRILLED IN THE AREA AND CALCULATE
C     THE CONSTANTS WHICH ARE REQUIRED FOR THE GALLE AND
C     WOODS EQUATIONS AS DESCRIBED IN SECTION III-B
C
C     THE INPUT VARIABLES ARE AS FOLLOWS
C
C     N - NUMBER OF STAGES IN BIT RUN
C      WW(J) - WEIGHT USED AT STAGE J
C      RR(J) - SPEED USED AT STAGE J
C      DELF(J) - DISTANCE DRILLED AT STAGE J
C      DELT(J) - TIME TO DRILL STAGE J
C      SDEPTH - DEPTH AT START OF BIT RUN
C      TDEPTH - DEPTH AT END OF BIT RUN
C      DF - FINAL TOOTH WEAR
C      BXF - FINAL BEARING W[illegible]
C      TF - TOTAL ROTATING TIME       [illegible]
C      FF - TOTAL DISTANCE DRILLED DURING BIT RUN
C      FORM - FORMATION RESPONSE
C              0. IF SOFT
C              1. IF HARD
C      FORMK - EXPONENT ON WEIGHT IN DRILLING RATE EQUATION
C      HOLE - HOLE SIZE
C        [illegible]
C          0. IF SS
C          1. IF FC
C          2. IF TC
C

      DIMENSION WW(10),RR(10),DELF(10),DELT(10),WEAR(3
     [illegible]
      [illegible]
      DATA RESP/'SOFT','HARD'/
       DULLA(D)= 1.0+(6.0+.928125*D)*D
      DULLAI(D)=(1.0+(3.+.309375*D)*D)*D
      DULLAH(D)=(1.0+(6.0+.928125*D)*D)**0.5
       [illegible]
     **D)*D
       KOUNT=10
      WRITE(6,600)
      WRITE(6,601)
   20 FORMAT(I2,3X,14F5.1,/,16F5.1,/,10F5.1)
   21 FORMAT(2F10.3,12F5.1)
   99 READ(5,20) N,(WW(J),RP(J),DELF(J),DELT(J),J=1,N)
      IF(N.EQ.0) STOP
       IERR=0
       [illegible]
      COM1=0.
```

```
C                    MAIN              ...(CONT'D)
      CON2=0.
      [illegible]
      [illegible]
      READ(5,21) SDEPTH,TDEPTH,DF,BXF,TF,FF,FORM,FORMK,HOLE
     *[illegible]
      [illegible]
      DO 50 J=1,N
      WEIGHT=WW(J)
      REVS=RR(J)
      A=EXP(-100./REVS**2)
      [illegible]
    1   ARR=A*REVS**0.75+0.5*REVS*(1.-A
      GO TO 3
    2   ARR=A*REVS**0.428+0.2*REVS*(1.-A)
    3 WBAR=7.88*WEIGHT/HOLE
      EMWATE=1359.1-714.19*ALOG10(WBAR)
      ROTI=REVS*(1.+.0004348*REVS**2)
      XLOAD=21335.87/(1.+.031147*WBAR)**3.2255
      X1=REVS/XLOAD
      X2=ROTI/EMWATE
      X3=(WBAR**FORMK)*ARR
     [illegible]
     [illegible]
      IF(IP.EQ.3) GO TO 30
      CON1=CON1+X2*DELT(J)
         [illegible]
        [illegible]
   30 CON2=CON2+DELF(J)/(X3*DELT(J))
   50 CONTINUE
      DFS=CON3/BXF
      GO TO (60,60,70),IP
   60 AF=CON1/DULLAI(DF)
      IF(IP.EQ.1) CF=CON2/(AF*SIN1(DF))
      IF(IP.EQ.2) CF=CON2/(AF*DF)
      GO TO 80
   70 CF=CON2/FLOAT(N)
    0 IF(ABS(TDEPTH-(SDEPTH+FEET)).GT.1.0) IERR=1
      IF(IERR.EQ.1) WRITE(6,603)
      IF(FORM.LT.0.5) IJK=1
      [illegible]
       WRITE(6,602) SDEPTH,TDEPTH,AF,CF,DFS,RESP(IJK),FORM
     *,WEAR(IP)
      KOUNT=KOUNT+2
       IF(KOUNT.LE.52) GO TO 90
      WRITE(6,600)
        [illegible]
      KOUNT=10
   90 GO TO 99
  600 FORMAT('1',5(/))
  601 FORMAT(14X,'INTERVAL       AF      CF      DFS    SPEED
```

```
      C                   MAIN            ...(CONT'D)

   * WEIGHT',
   1'    MEAR',/,45X,'RESPONSE  RESPONSE  TYPE',/)
602 FORMAT(11X,F6.0,'-',F6.0,1X,F6.2,1X,F6.4,1X,F5.2,3X,A4
   *,6X,F4.2,5X
   1,A2)
603 FORMAT(//,22X,'POSSIBLE ERROR',/,22X,'DEPTHS
   * INCONSISTENT',//)
    END
```

```
        EXAMPLE OF THE PRINT-OUT FROM THE CONSTANTS PROGRAM

    INTERVAL          AF      CF        S       SPEED       FLIGHT     BEAM
                                               RESPONSE    RESPONSE    TYPE

[illegible]
12400.-13000.        .00    .00500    2.00     HARD         1.00        T[illegible]
[illegible]          4.22   .00514    1.0[illegible]     HARD         1.0[illegible]        SO
```

Appendix E

Mobil Oil Test Data

```
                        GROUP NO.  1

        BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
        (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
                    1.800  0.50/1000FT    2.00       SS

                   INTERVAL  1400. - 2920.

  AF     CF       S     SPEED    WEIGHT          CONSTRAINTS
                      RESPONSE RESPONSE   W MIN  W MAX  R MIN  R MAX
5.00 0.0450  2.00    SOFT       1.00       30.    60.    40.   150.

                   INTERVAL  2920. - 3770.

  AF     CF       S     SPEED    WEIGHT          CONSTRAINTS
                      RESPONSE RESPONSE   W MIN  W MAX  R MIN  R MAX
4.50 0.0400  1.80    SOFT       1.00       30.    60.    40.   140.

                   INTERVAL  3770. - 5070.

  AF     CF       S     SPEED    WEIGHT          CONSTRAINTS
                      RESPONSE RESPONSE   W MIN  W MAX  R MIN  R MAX
4.00 0.0350  1.70    SOFT       1.00       30.    60.    40.   140.

                   INTERVAL  5070. - 6770.

  AF     CF       S     SPEED    WEIGHT          CONSTRAINTS
                      RESPONSE RESPONSE   W MIN  W MAX  R MIN  R MAX
4.50 0.0400  2.00    SOFT       1.00       30.    50.    40.   140.


                        GROUP NO.  2

        BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
        (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
         3.750


  AF     CF       S     SPEED    WEIGHT          CONSTRAINTS
                      RESPONSE RESPONSE   W MIN  W MAX  R MIN  R MAX
2.00 0.0400  1.50    HARD       1.00       25.    50.    30.    60.


                        GROUP NO.  3

        BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
        (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
         3.750      1.560  0.50/1000FT    2.00       SS
```

```
                    INTERVAL  6310. -  8020.

  AF      CF        S     SPEED     WEIGHT           CONSTRAINTS
                        RESPONSE RESPONSE   WMIN   WMAX   RMIN   RMAX
3.00  0.0400   1.80    SOFT       1.00       30.    60.    40.   110.


                         GROUP NO.  4

     BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
     (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
       8.750     9.300  0.50/1000FT     2.50       SS

                    INTERVAL  8020. -  8260.

  AF      CF        S     SPEED     WEIGHT           CONSTRAINTS
                        RESPONSE RESPONSE   WMIN   WMAX   RMIN   RMAX
2.00  0.0400   1.50    HARD       1.00       25.    50.    30.    60.


                         GROUP NO.  5

     BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
     (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
       8.750     1.860  0.50/1000FT     2.50       SS

                    INTERVAL  8260. -  9020.

  AF      CF        S     SPEED     WEIGHT           CONSTRAINTS
                        RESPONSE RESPONSE   WMIN   WMAX   RMIN   RMAX
3.00  0.0400   1.80    SOFT       1.00       30.    60.    40.   100.


                         GROUP NO.  6

     BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
     (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
       8.750     9.300  0.50/1000FT     2.50       SS

                    INTERVAL  9020. -  9600.

  AF      CF        S     SPEED     WEIGHT           CONSTRAINTS
                        RESPONSE RESPONSE   WMIN   WMAX   RMIN   RMAX
2.00  0.0400   1.50    HARD       1.00       25.    50.    30.    60.


                         GROUP NO.  7

     BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
     (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
       8.750     1.860  0.50/1000FT     2.50       SS
```

```
                    INTERVAL  9060. -  9270.

 AF      CF       S     SPEED     WEIGHT            CONSTRAINTS
                      RESPONSE RESPONSE   WMIN   WMAX   RMIN   RMAX
3.00 0.0400  1.00    SOFT      1.000      30.    60.    40.    50.

                    INTERVAL  9270. - 10060.

 AF      CF       S     SPEED     WEIGHT            CONSTRAINTS
                      RESPONSE RESPONSE   WMIN   WMAX   RMIN   RMAX
6.00 0.0450  2.00    SOFT      1.000      30.    60.    40.    [illegible]


                         GROUP NO. 8

      BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
      (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
       8.750      9.300  0.50/1000FT     2.50       SS

                    INTERVAL 10060. - 10750.

 AF      CF       S     SPEED     WEIGHT            CONSTRAINTS
                      RESPONSE RESPONSE   WMIN   WMAX   RMIN
10.00 0.0500  2.00    HARD      1.000      30.    45.    35.

                    INTERVAL 10750. - 11313.

 AF      CF       S     SPEED     WEIGHT            CONSTRAINTS
                      RESPONSE RESPONSE   WMIN   WMAX   RMIN   RMAX
1.50 0.0400  1.50    HARD      1.000      30.    45.    35.    70.


                         GROUP NO. 9

      BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
      (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
       8.750      1.250  0.50/1000FT     2.50       SS

                    INTERVAL 11313. - 11680.

 AF      CF       S     SPEED     WEIGHT            CONSTRAINTS
                      RESPONSE RESPONSE   WMIN   WMAX   RMIN   RMAX
9.00 0.0400  2.00    SOFT      1.000      30.    45.    25.    70.


                         GROUP NO. 10

      BIT SIZE  BIT COST  TRIP RATE  CHANGE TIME  WEAR
      (INCHES)   (HOURS)   (HOURS)     (HOURS)    TYPE
       8.750      9.300  0.50/1000FT     2.50       SS
```

```
                   INTERVAL 11420. - 12084.

  AF      CF       S    SPEED     WEIGHT            CONSTRAINTS
                       RESPONSE RESPONSE    MIN   MAX   RMIN  RMAX
5.00  0.0400   2.00    HARD      1.00      30.   45.   35.

                   INTERVAL 12084. - 12420.

  AF      CF       S    SPEED     WEIGHT            CONSTRAINTS
                       RESPONSE RESPONSE    MIN   MAX   RMIN  RMAX
5.00  0.0510   2.00    HARD      1.00      30.   45.   35.   70.

                   INTERVAL 12420. - 13216.

  AF      CF       S    SPEED     WEIGHT            CONSTRAINTS
                       RESPONSE RESPONSE    MIN   MAX   RMIN  RMAX
2.00  0.0500   2.00    HARD      1.00      30.   45.   35.   70.

                   INTERVAL 13216. -

  AF      CF       S    SPEED     WEIGHT            CONSTRAINTS
                       RESPONSE RESPONSE    MIN   MAX   RMIN  RMAX
2.00  0.0670   2.00    HARD      1.00      30.   45.   35.
```

Appendix F

Program and Results of "Point" Optimization

```
      C                MAIN
C           MAIN
C
C     THIS PROGRAM OPTIMISES THE DRILLING PROCEDURE USING
C    THE METHOD OF ''POINT'' OPTIMIZATION.
C

      DIMENSION HOW(4)
      DIMENSION RESP(2),WEAR(3)
      COMMON AF,CF,DFS,FORM,FORMK,P,ENR
      COMMON HOLE,BITHR,TRIPT,FOOTIN,FEET,DOUT,HOUT,REPTIM
      COMMON DIN,RIN,IP,XPAR,DELTAT,DELTAF,FIB,IFIB(30)
      COMMON INDIC1,INDIC2,INDIC3,AINC1,AINC2
      COMMON XLIMLO(2),XLIMHI(2),CORR(2)
      COMMON WBEST,RBEST,DETRES,DEFRES,DBEST,SBEST
      COMMON FOOT1,TIME1,IOPT4
      DATA WEAR/'SS','FC','TC'/
      DATA RESP/'SOFT','HARD'/
      DATA HOW/'    ','    ',' (CN','TD.)'/
    1 FORMAT(8F10.3)
    2 FORMAT(40I2)
    3 FORMAT(5F10.3,6F5.1)
 6000 FORMAT('1',///,52X,'SUMMARY OF INPUT DATA',/)
 6001 FORMAT(5X,'INTERVAL          AF     CF       S     SPEED',
     1'     WEIGHT            CONSTRAINTS          BIT SIZE  BIT'
     2,' COST  TRIP RATE  CHANGE TIME  WEAR',/,37X,
     3'RESPONSE RESPONSE  W IN  WMAX  RMIN  RMAX  (INCHES)'
     4,'    (HOURS)    (HOURS)       (HOURS)     TYPE',/)
 6003 FORMAT(53X,'P R O C E D U R E',// )
[illegible]
[illegible]
[illegible]
 6020 FORMAT(27X,'BIT NO.    INTERVAL     WEIGHT  SPEED    '
     1,'ROT. TIME  TRIP TIME  EQUIV. COST  WEAR',/,97X,
     2'(RIG HOURS)  T  B',/)
 6006 FORMAT(29X,I2,2X,F7.1,'--',F7.1,2X,F5.0,4X,F4.0,3X,
     1F7.2,4X,F7.2,5X,F7.2,5X,I1,2X,I1,//)
 6007 FORMAT(39X,'FORMATION ROTATING TIME   = ',F7.2,
     1' RIG HOURS',//,
     1          39X,'FORMATION TRIP TIME       = ',F7.2,
     1' RIG HOURS',//,
     1          39X,'FORMATION RIG TIME        = ',F7.2,
     1' RIG HOURS',//,
     2          39X,'FORMATION COST            = ',F7.2,
     1' RIG HOURS',//,
     2          39X,'FORMATION BIT COUNT       = ',3X,I2,//)
 6008 FORMAT(39X,'CUMULATIVE ROTATING TIME = ',F7.2,
     1' RIG HOURS',//,
     1          39X,'CUMULATIVE TRIP TIME      = ',F7.2,
     1' RIG HOURS',//,
     1          39X,'CUMULATIVE RIG TIME       = ',F7.2,
```

```
    C                  MAIN          ...(CONT'D)

      1' RIG HOURS',//,
      2         39X,'CUMULATIVE COST                   = ',F7.2,
      1' [illegible]
      2         39X,'CUMULATIVE BIT COUNT              = ',3X,I2,//)
C     THE INPUT VARIABLES ARE AS FOLLOWS
C      ERR - STEP SIZE AT WHICH SEARCH IS STOPPED - TRY 0.1
C      AINC1 - WEIGHT INCREMENT FOR CALCULATING DERIVATIVE
C               TRY 0.5
C      AINC2 - SPEED INCREMENT FOR CALCULATING THE DERIVATIVE
C                    1.
C      CORR(1) - WEIGHTING FACTOR - TRY 1.
C      CORR(2) - WEIGHTING FACTOR - TRY 4.
C      NINT - NO. OF SECTIONS IN GROUP
C      NUSET - SWITCH - ZERO UNTIL THE LAST DATA SET WHEN A
C               NON-ZERO NUMBER IS NEEDED
C      HOLE - HOLE SIZE
C      BITCST - BIT COST
C      RIGCST - RIG COST
C      TRIPT - TRIP RATE - TIME PER RETURN FOOT
C      REPTIM - CONNECTION COST
C      P - WEAR TYPE
C          [illegible]
C          1. IF IF FC
C          2. IF TC
C      SDEPTH - DEPTH AT TOP OF SECTION
C      TDEPTH - DEPTH AT BOTTOM OF SECTION
C      AF - CONSTANT IN DULLING EQUATION
C      CF - CONSTANT IN THE DRILLING RATE EQUATION
C      DFS - CONSTANT IN THE BEARING WEAR EQUATION
C      FORM - RESPONSE TO SPEED, 0. IF SOFT, 1. IF HARD
C      FORMK - EXPONENT ON WEIGHT
C      XLIMLO( ) - LOWER LIMITS ON WEIGHT AND SPEED
C      XLIMHI( ) - UPPER LIMITS ON WEIGHT AND SPEED
C
C
C
C     IMPORTANT INTERNAL VARIABLES ARE AS FOLLOWS
C
C      DIN - INITIAL TOOTH WEAR
C      DOUT - FINAL TOOTH WEAR
C      BIN - INITIAL BEARING WEAR
C      BOUT - FINAL BEARING WEAR
C      ICBIT - CUMULATIVE NUMBER OF BITS
C      CTROT - CUMULATIVE ROTATING TIME
C      CTRIP - CUMULATIVE TRIP TIME
C      CEQ - CUMULATIVE COST
C      STROT - ROTATING TIME FOR SECTIONS IN A GROUP
C      STRIP - TRIP TIME FOR SECTIONS IN A GROUP
C      STEQ - COST OF SECTIONS IN A GROUP
```

```
      C                  MAIN           ...(CONT'D)

      ICBIT=0
      CTROT=0.
      [illegible]
      [illegible]
      [illegible]
      READ(5,1) ERR,AINC1,AINC2,(CONR(J),J=1,2)
      IFIB(1)=1
      [illegible]=1
      [illegible]
      [illegible]
   [illegible]
      FIB=FLOAT(IFIB(IOPT4))
   99 READ(5,2) NINT,NUSET
      STROT=0.
      STRIP=0.
      STEQ=0.
      DIM=0.
      BIN=0.
      INDIC3=0
      NBIT=0
      [illegible]
      TIME1=0.
      DO 10 JJ=1,NINT
      READ(5,1) HOLE,BITCST,RIGCST,TRIPT,REPTIM,P
      [illegible]
      XPAR=7.38/HOLE
      BITHR=BITCST/RIGCST
      [illegible]
      READ(5,3) SDEPTH,TDEPTH,AF,CF,DFS,FORMN,FORMK,(XLIMLO
     1(J),J=1,2),(XLIMHI(J),J=1,2)
      [illegible]
      WRITE(6,6000)
      [illegible]
      WRITE(6,6002) SDEPTH,TDEPTH,AF,CF,DFS,RESP(IJK),FORMN,
     1XLIMLO(1),XLIMHI(1),XLIMLO(2),XLIMHI(2),HOLE,BITHR,
     1TRIPHR,REPTIM,WEAR(IF)
      [illegible]
      WRITE(6,6020)
      FEET=SDEPTH
   11 CALL ASCENT
      FTEMP=FEET+DEFRES
      IF(FTEMP.GT.TDEPTH) GO TO 20
      IF(INDIC3.EQ.1) TRIP=(TRIPT*FTEMP+REPTIM)/2.
      IF(INDIC3.EQ.0) TRIP=TRIPT*(FEET+FTEMP)/2.+REPTIM
      TEQ=DETRES+TRIP+BITHR
      STROT=STROT+DETRES
      STRIP=STRIP+TRIP
      STEQ=STEQ+TEQ
```

```
C                   MAIN            ...(CONT'D)

      ID=IFIX(8.*DBEST+0.5)
      IB=IFIX(8.*BBEST+0.5)
      SRIG=STROT+STRIP
      IF(INDIC3.EQ.0) NBIT=NBIT+1
      WRITE(6,6006) NBIT,FEET,FTEMP,WBEST,RBEST,DETRES,
     [illegible]
      FEET=FTEMP
      DIN=0.
      BIN=0.
      INDIC3=0
      [illegible]
      [illegible]
      GO TO 11
[illegible] FOOTIN=TDEPTH-FEET
      INDIC2=1
      CALL TEST(WBEST,RBEST,DCOST)
      DBEST=DOUT
      BBEST=BOUT
      DETRES=DELTAT
      FTEMP=TDEPTH
      TRIP=0.
       IF(INDIC3.EQ.0) TRIP=(TRIPT*FEET+REPTIM)/2.
      IF(JJ.EQ.NINT) TRIP=TRIP+(TRIPT*TDEPTH+REPTIM)/2.
      TEQ=DETBES+TRIP+BITHR
      STROT=STROT+DETBES
      STRIP=STRIP+TRIP
      [illegible]
      ID=IFIX(8.*DBEST+0.5)
      [illegible]
      SRIG=STROT+STRIP
      IF(INDIC3.EQ.0) NBIT=NBIT+1
      WRITE(6,6006) NBIT,FEET,FTEMP,WBEST,RBEST,DETBES,
     1TRIP,TEQ,ID,IB
      INDIC3=1
      DIN=DBEST
      BIN=BBEST
      [illegible]
      TIME1=TIME1+DETBES
[illegible] CONTINUE
      [illegible]
      ICBIT=ICBIT+NBIT
      [illegible]
      CTRIP=CTRIP+STRIP
      CEQ=CEQ+STEQ
      CRIG=CRIG+SRIG
      WRITE(6,6008) CTROT,CTRIP,CRIG,CEQ,ICBIT
      IF(NUSET.EQ.0) GO TO 99
      STOP
      END
```

SUBROUTINE ASCENT

```
      SUBROUTINE ASCENT
C
C     THIS SUBROUTINE USES A GRADIENT SEARCH METHOD TO GET
C     THE OPTIMUM SPEED AND WEIGHT.  NORMALLY THE SEARCH
C     TERMINATES AFTER THE STEP SIZE BECOMES SMALLER THAN ERR
C     BUT IF THE SEARCH REACHES A CORNER OF THE PERMITTED
C     AREA IT IS TERMINATED FOR THAT REASON.  THE SUBROUTINE
C     ALSO CONDUCTS A SEARCH ON THE FINAL WEAR CONDITION
C     IF THE BIT IS FOUND TO BE RATE CONTROLLED.  THE START-
C     ING VALUES FOR THE SEARCH ARE THE UPPER LIMITS ON THE
C     WEIGHT AND SPEED
C
C
C
C     INTERNAL VARIABLES ARE AS FOLLOWS
C      SS - STEP SIZE
C      FBEST - BEST VALUE OF FUNCTION FOUND AT ANY STAGE
C      FX - FUNCTION VALUE
C      XL1(1) - DERIVATIVE W.R.T. WEIGHT
C      XL1(2) - DERIVATIVE W.R.T. SPEED
C      X1( ) - BASE VALUES FROM WHICH STEPS ARE MADE
C      X2( ) - POINTS AT WHICH FUNCTION IS EVALUATED
C      DETBES - BEST VALUE OF ROTATING TIME
C      DEFBES - BEST VALUE OF FOOTAGE
C      DBEST - VALUE OF FINAL TOOTH WEAR ASSOCIATED WITH THE
C              BEST WEIGHT AND SPEED
C      BBEST - VALUE OF FINAL BEARING WEAR ASSOCIATED WITH
C              THE BEST WEIGHT AND SPEED
C
      DIMENSION X1(2),X2(2),XL1(2)
      LOGICAL A,B,C,D
      COMMON AF,CF,DFS,FUP[illegible],FOR[illegible]K,P,ERR
      COMMON HOLE,BIT [illegible],TRIPT,FOOTIN,FEET,DOUT,BOUT,REFTIN
      COMMON DIN,SIN,IP,XPAR,DELTAT,DELTAF,FIB,IFIB(30)
      COMMON INDIC1,INDIC2,INDIC3,AINC1,AINC2
      COMMON XLIMLO(2),XLIMHI(2),C[illegible]R(2)
      COMMON WBEST,XBEST,DETBES,DEFBES,D[illegible]EST,FBEST
      COMMON FOOT1,TIME1,IOPT4
      INDIC2=0
      INDIC1=
      [illegible]
      X1(2)=XLIMHI(2)
   10 SS=1.
      LL=0
      KL=0
      CALL TEST(X1(1),X1(2),FBEST)
    9 CALL DERIV(X1(1),X1(2),XL1,FBEST)
      [illegible]
    1 GO TO 100
    5 [illegible]
```

SUBROUTINE ASCENT ...(CONT'D)

```
      X2(J)=X1(J)-SS*XL1(J)*CORR(J)
      IF(X2(J).GT.XLIMHI(J)) X2(J)=XLIMHI(J)
    4 IF(X2(J).LT.XLI[illegible]XLIMLO(J)
      CALL TEST(X2(1),X2(2),FX)
      IF(FX.LT.FBEST)  GO TO 6
      IF([illegible].LT.[illegible])  GO TO 100
      SS=SS/2.
      LL=LL+1
      GO TO 5
    6 FBEST=FX
      KL=KL+1
      IF(LL.GT.0) GO TO 30
      SS=SS*2.
      DO 7 J=1,2
      X2(J)=X1(J)-SS*XL1(J)*CORR(J)
      IF(X2(J).GT.XLIMHI(J)) X2(J)=XLIMHI(J)
    7 IF(X2(J).LT.XLIMLO(J)) X2(J)=XLIMLO(J)
      CALL TEST(X2(1),X2(2),FX)
      IF(FX.LT.FBEST)  GO TO 6
      SS=SS/2.
   30 DO 8 J=1,2
      X1(J)=X1(J)-SS*XL1(J)*CORR(J)
      IF(X1(J).GT.XLIMHI(J)) X1(J)=XLIMHI(J)
    8 IF(X1(J).LT.XLIMLO(J)) X1(J)=XLIMLO(J)
      A=ABS(X1(1)-XLIMHI(1)).LT..1
      B=ABS(X1(2)-XLIMHI(2)).LT..1
      IF(A.AND.B) GO TO 100
      C=ABS(X1(2)-XLIMLO(2)).LT..1
      IF(A.AND.C) GO TO 100
      D=ABS(X1(1)-XLIMLO(1)).LT..1
      IF(D.AND.B) GO TO 100
      [illegible]
      LL=0
      [illegible]
      GO TO 9
  100 IF(INDIC1.EQ.1)  GO TO 101
      CALL TEST([illegible])
      WBEST=X1(1)
      [illegible]T=X1(2)
      TBEST=FBEST
      DETBES=DELTAT
      DEFBES=DELTAF
      DBEST=DOUT
      BBEST=BOUT
      IF(INDIC1.EQ.2) RETURN
      DIFF=DOUT-DIN
      IF(DIFF.LT.0.06) RETURN
      DOUT=DOUT-0.05
      INDIC1=1
      KNT=0
```

```
      GO TO 10
[illegible] KNT=KNT+1
      IF(KNT.NE.1)  GO TO 102
      [illegible]
      RANGE=DBEST-DIN
      A1=DIN
      A3=DBEST
      DEL=RANGE*FLOAT(IFIB(IOPT4-2))/FIB
      DOUT=A1+DEL
      XY1=DOUT
      [illegible]
      GO TO 10
102   [illegible]
110   FX1=FBEST
      [illegible]
      XY1=DOUT
      JJ=2
      GO TO 10
111   FY1=FBEST
      [illegible]
      KKK=IOPT4-2
      DO 120 I=1,KKK
      LL=KKK-I
      DEL=RANGE*FLOAT(IF[illegible]
      IF(LL.EQ.1) DEL=DE[illegible]
      [illegible] TO 123
      A3=XY1
      [illegible]
      FY1=FX1
      [illegible]
      DOUT=XX1
      JJ=3
      GO TO 10
112   FX1=FBEST
      GO TO 120
[illegible] A1=XX1
      XX1=XY1
      FX1=FY1
      [illegible]
      DOUT=XY1
      JJ=4
      GO TO 10
113   FY1=FBEST
120   CONTINUE
      INDIC1=2
      IF(FX1.GT.FY1) GO TO 136
      DOUT=XX1
      GO TO 10
136   DOUT=XY1
      GO TO 10
```

SUBROUTINE ASCENT ...(CONT'D)

```
      SUBROUTINE TEST(WEIGHT,REVS,RCOST)

     THIS SUBPROGRAM USES THE GIVEN VALUE OF WEIGHT AND
     SPEED PLUS THE INITIAL WEAR CONDITIONS PRESENT AT THE
     TIME TO DO ONE OF TWO THINGS.  EITHER IT CALCULATES THE
     FINAL WEAR CONDITIONS AND THE FOOTAGE DRILLED, THE TIME
     TAKEN AND THE COST PER FOOT, OR, IF THE SWITCHES ARE
     SET IN THE OTHER POSITION IT CALCULATES THE TIME,
     COST AND FINAL WEAR CONDITIONS FOR A GIVEN FOOTAGE


      COMMON AF,CF,DFS,FORM,FORMK,P,FPP
      COMMON HOLE,BITHR,TRIPT,FOOTIN,FEET,DOUT,BOUT,REPTI.
      COMMON DIN,BIN,IP,XBAR,DELTAT,DELTAF,FIR,IFIR(30)
      COMMON INDIC1,INDIC2,INDIC3,AINC1,AINC2
      COMMON XLIMLO(2),XLIMHI(2),CBAR(2)
      COMMON WBEST,RBEST,RETRES,WTRIES,PBEST,WBEST
      COMMON FOOT1,TIME1,INPT4
       DULLA(D)= 1.0+(6.0+.92*125*D)*D
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      IP=IFIX(P/0.9)+1
      [illegible]
      A=EXP(-100./REVS**2)
      IF(FORM-0.5) 1,1,2
 1      ARR=A*REVS**0.75+0.5*REVS*(1.-A)
      GO TO 3
        [illegible]
      WBAR=7.88*WEIGHT/HOLE
      EMWATE=1359.1-714.19*ALOG10(WBAR)
      [illegible]
      XLOAD=21335.87/(1.+.031167*WBAR)**3.225
      F1=AF*EMWATE/ROTI
      F2=F1*CF*(WBAR**FORMK)*ARR
      F3=F1*REVS/DFS/XLOAD
      IF(INDIC2.EQ.1)  GO TO 111
      GO TO (18,18,28),IP
 18   IF(INDIC1.NE.0)  GO TO 101
      DOUT=1.
 101  BOUT=BIN+(DULLAI(DOUT)-DULLAI(DIN))*F3
      IF(BOUT.LE.1)  GO TO 5
      [illegible]
      C=DULLAI(DIN)+(1.-BIN)/F3
      X=1.
      DOUT=X-(DULLAI(X)-C)/DULLA(X)
      IF(ABS((DOUT-X)/DOUT).LT.1.E-05)  GO TO 5
      [illegible]
      GO TO 4
```

SUBROUTINE TEST ...(C[illegible])

```
  5 DELTAT=F1*(DULLAI(DOUT)-DULLAI(DIN))
    GO TO (51,52,28),IP
 52 DELTAF=F2*(DOUT-DIN)
    GO TO 48
 51 DELTAF=F2*(SIM1(DOUT)-SIM1(DIN))
 48 FOOT2=FOOT1+DELTAF
    DCOST=(BITHR+REPTIM+TRIPT*(FEET+(DELTAF-FOOT1)/2.)
   [illegible]
    RETURN
111 CONTINUE
    GO TO (19,29,39),IP
 19 X=DIN+0.2
    C=SIM1(DIN)+FOOTIN/F2
  9 DOUT=X-(SIM1(X)-C)/DULLAH(X)
    IF(ABS((DOUT-X)/DOUT).LT.1.E-05[illegible]
    [illegible]=DOUT
    GO TO 9
 10 DIFF=DULLAI(DOUT)-DULLAI(DIN)
    BOUT=BIN+F3*DIFF
    [illegible]
 49 FOOT2=FOOT1+FOOTIN
    DCOST=(BITHR+REPTIM+TRIPT*(FEET+(FOOTIN-FOOT1)/2.)
   1+DELTAT+TIME1)/FOOT2
    RETURN
 39 F1=DFS*XLOAD/REVS
    BOUT=FOOTIN/(F1*C[illegible]WBAR**FORMK)
    [illegible]
    [illegible]
    GO TO 49
[illegible] DOUT=FOOTIN/F2+DIN
    DDIFF=DULLAI(DOUT)-DULLAI(DIN)
    BOUT=F3*DDIFF
    DELTAT=F1*DDIFF
    GO TO 49
 28 DELTAT=F1/F3*(1.-BIN)
    DELTAF=F2/F3*(1.[illegible]
    [illegible]
    GO TO 48
    END
```

```
RESULTS OF SINGLE [illegible] [illegible]      C[illegible] I.
CONSTANTS IMPLIED [illegible]        .


                    GROUP NO.  1

    PIT NO.              INTERVAL    WEIGHT   SPEED

                    1406. -  1031.   60.      153.
                    1031. -  2200.   59.      150.
                    2200. -  2700.   61.      144.
                    2700. -  2920.   6[illegible].      146.

                    GROUP NO.  2

    PIT NO.              INTERVAL    WEIGHT   SPEED

      1             2920. -  3223.   60.      140.
      2             3223. -  3577.   60.      153.
      3             3577. -  3770.   60.      157.

                    GROUP NO.  3

    PIT NO.              INTERVAL    WEIGHT   SPEED

      1             3770. -  4030.   59.      132.
      2             4030. -  4425.   59.      132.
      3             4425. -  4730.   59.      125.
      4             4730. -  5058.   59.      125.
      5             5058. -  5170.   59.      127.

                    GROUP NO.  4

    PIT NO.              INTERVAL    WEIGHT   SPEED

      1             5170. -  5476.   60.      134.
      2             5476. -  5844.   59.      137.
      3             5844. -  6257.   60.      130.
      4             6257. -  6717.   59.      125.
      5             6717. -  6770.   59.      121.

                    GROUP NO.  5

    PIT NO.              INTERVAL    WEIGHT   SPEED

      1                 [illegible]          50.       60.
```

```
                    GROUP NO.  6

BIT NO.                 INTERVAL    WEIGHT  SPEED

   1               6810. -  7150.   55.     110.
   2               7150. -  7491.   56.     110.
   3               7491. -  7834.   56.     110.
   4               7834. -  8180.   57.     110.
   5               8180. -  8527.   57.     109.
   6               8527. -  8820.   57.     109.

                    GROUP NO.  7

BIT NO.                 INTERVAL    WEIGHT  SPEED

   1               8820. -  8860.   50.      40.

                    GROUP NO.  8

BIT NO.                 INTERVAL    WEIGHT  SPEED

   1               8860. -  9020.   53.     100.

                    GROUP NO.  9

BIT NO.                 INTERVAL    WEIGHT  SPEED

   1               9020. -  9060.   50.      40.


                    GROUP NO. 10

BIT NO.                 INTERVAL    WEIGHT  SPEED

   1               9060. -  9270.   50.      90.

                    GROUP NO. 11

BIT NO.                 INTERVAL    WEIGHT  SPEED

   1               9270. -  9960.   60.      90.
   2               9960. - 10080.   60.      90.

                    GROUP NO. 12

BIT NO.                 INTERVAL    WEIGHT  SPEED

   1              10080. - 10401.   45.      70.
   2              10401. - 10721.   45.      70.
```

```
                    GROUP NO. 13

BIT NO.              INTERVAL    WEIGHT  SPEED

   1            [illegible] - 10869.  45.    [illegible]
   2            10869. - 10988.  45.    [illegible]
   3            10988. - 11107.  45.    [illegible]
   4            11107. - 11226.  45.    [illegible]
   5            11226. - 11313.  45.    [illegible]

                    GROUP NO. 14

BIT NO.              INTERVAL    WEIGHT  [illegible]

   1            11313. - 11690.  45.    [illegible]

                    GROUP NO. 15

[illegible]          INTERVAL    WEIGHT  [illegible]

   1            11690. - 11899.  45.    [illegible]
                11899. - 12094.  45.

                    GROUP NO. 16

[illegible]          INTERVAL    WEIGHT  [illegible]

   1            12094. - 12354.  45.
   2            12354. - 12632.  45.
   3            12632. - 12917.  45.
   4            12917. - 12923.  45.

                    GROUP NO. 17

[illegible]          INTERVAL    WEIGHT  [illegible]

   1            12923. - 13098.  45.
                13098. - 13216.  45.

                    GROUP NO. 18

[illegible]          INTERVAL    WEIGHT  [illegible]

                13216. - 13421.  45.
                13421. - [illegible]  45.

CUMULATIVE ROTATING TIME = [illegible].21 RIG HOURS
CUMULATIVE [illegible] TIME  =  226.04 RIG HOURS
CUMULATIVE [illegible] TIME  = 1126.25 RIG HOURS
CUMULATIVE [illegible]       = [illegible] RIG HOURS
CUMULATIVE [illegible] COUNT = [illegible]
```

```
RESULTS OF MODIFIED [illegible] ([illegible]). THE GROUPS
IN THIS CASE ARE THE SAME AS SUGGESTED BY [illegible].

                         GROUP NO.  1

  IT NO.                   INTERVAL          WEIGHT    SPEED

   1                  14[illegible]. -  1831.     60.      150.
   2                  1831. -  2263.     [illegible]      1[illegible]0.
   3                  2263. -  27[illegible].               140.
   4                  27[illegible]. -  2920.     6[illegible].      140.
   4 (CNTD.)          2920. -  31[illegible].     60.      140.
   5                  31[illegible]. -  3406.     [illegible]      1[illegible].
   6                  34[illegible]. -  3770.     60.      1[illegible].
   6 ([illegible])    3770. -  3[illegible]4[illegible].     60.      140.
   7                  3[illegible]. -  4164.     59.      132.
   8                  4164. -  4[illegible]4.     59.      132.
   9                  44[illegible]4. -  4810.     59.      12[illegible].
  10                  4810. -  5[illegible]7[illegible].               12[illegible].
  [illegible] (CNTD.)     5070. -  5149.     60.      140.
  [illegible]                 5149. -  5534.     5[illegible].
  [illegible]                 5534. -  5763.     5[illegible].      132.
  [illegible]                                     50.      120.
  [illegible]                 6[illegible]. -  6770.     59.      120.

                         GROUP NO.  2

  [illegible]                  INTERVAL          WEIGHT

   1                  6770. -  6810.

                         GROUP NO.  3

  [illegible]                  INTERVAL          WEIGHT    SPEED

   1                  6810. -  7150.               110.
   2                  7150. -  7491.     58.      110.
   3                  7491. -  7834.     5[illegible].      110.
   4                  7834. -  8180.               110.
   5                  8180. -  8527.               10[illegible].
   6                  8527. -  8820.     57.      10[illegible].

                         GROUP NO.  4

  IT NO.                   INTERVAL          WEIGHT    SPEED

   1                  8820. -  8860.     50.       46.
```

```
                         GROUP NO.  5

BIT NO.                    INTERVAL    WEIGHT   SPEED

   1                     8800. -  9020.  50.      100.

                         GROUP NO.  6

BIT NO.                    INTERVAL    WEIGHT   SPEED

   1                     [illegible]

                         [illegible]

BIT NO.                    INTERVAL    WEIGHT   SPEED

   1                      9060. -  9270.  55.       90.
   1 (CNTD.)              9270. -  9635.  60.       90.
   2                      9635. - 10090.  60.       90.

                         GROUP NO.  8

BIT NO.                    INTERVAL    WEIGHT   SPEED

   1                     10090. - 10401.  45.       70.
   2                     10401. - 10721.  45.       70.
   3                     10721. - 10750.  45.       70.
   3 (CNTD.)             10750. - 10834.  45.       57.
   4                     10834. - 10957.  45.       41.
   5                     10957. - 11076.  45.       41.
   6                     11076. - 11195.  45.       41.
   7                     11195. - 11313.  45.       41.

                         GROUP NO.  9

BIT NO.                    INTERVAL    WEIGHT   SPEED

   1                     11313. - 11680.  45.       70.

                         GROUP NO. 10

BIT NO.                    INTERVAL    WEIGHT   SPEED

   1                     11680. - 11899.  45.       70.
   2                     11899. - 12084.  45.       70.
   2 (CNTD.)             12084. - 12127.  45.       70.
   3                     12127. - 12402.  45.       70.
   4                     12402. - 12676.  45.       70.
   5                     12676. - 12920.  45.       70.
   5 (CNTD.)             12920. - 12949.  45.       70.
```

```
BIT NUM             INTERVAL      WEIGHT

                 12949. -          -  45.       47.
                 .      - 15216.    45.
        (CNTD.)  15216. -          -  45.       35.
                 15512. -             45.       27.

CUMULATIVE ROTATING TIME  =   669.57 RIG HOURS
CUMULATIVE TRIP TIME      =   278.00 RIG HOURS
CUMULATIVE RIG TIME       =  1017.04 RIG HOURS
CUMULATIVE COST           =  1056.81 RIG HOURS
CUMULATIVE BIT COUNT          42
```

Appendix G

Program and Results of "Interval" Optimization

```
        C               MAIN

              MAIN

C     THIS PROGRAM OPTIMIZES THE DRILLING PROCEDURE USING
C     METHOD OF 'INTERVAL OPTIMIZATION' DESCRIBED IN SECTION
C     III-D.  THREE VARIATIONS ARE AVAILABLE FOR USE.
C      1)  EACH BIT DRILLS THE SAME DISTANCE
C      2)  THE DISTANCE DRILLED BY CONSECUTIVE BITS INCREASES
C          BY A CONSTANT INCREMENT.  THE VALUE OF THIS INCRE-
C          MENT IS FOUND BY FIBONACCI SEARCH.
C      3)  IDENTICAL WITH  (2) EXCEPT THAT ALL INCREMENTS ARE
C          TRIED IN CASE THE FUNCTION IS NOT UNIMODAL.
C

      DIMENSION IFIB(30)
       DIMENSION RESP(2) ,WEAR(3)
      DIMENSION F(2,100),WW(2,100),RR(2,100),TIME(2,100)
      DIMENSION ID(2,100),IB(2,100)
      COMMON AF,CF,DFS,FORM,FORMK
      COMMON TDEPTH,SDEPTH,HOLL,RIGCST,BITCST,TRIPT,REPTIM
      COMMON NO1,NOMAX,IW,IR
      COMMON XLO(2),XHI(2),ERR,A(550),R(1400),P
      DATA RESP/'SOFT','HARD'/
      [illegible]
    1 FORMAT(8F10.3)
    2 FORMAT(40I2)
    3 FORMAT(5F10.3,6F5.1)
 6000 FORMAT('1',///,52X,'SUMMARY OF INPUT DATA',/)
 6001 FORMAT(5X,'INTERVAL          AF     CF        S    SPEED',
     1'    WEIGHT             CONSTRAINTS          BIT SIZE  BIT'
     2,' COST  TRIP RATE  CHANGE TIME  WEAR',/,37X,
     3'RESPONSE RESPONSE  WMIN  WMAX  RMIN  RMAX  (INCHES)'
     4,'   (HOURS)   (HOURS)        (HOURS)     TYPE',/)
 6003 FORMAT(53X,'P R O C E D U R E',// )
 6002 FORMAT(2X,F6.0,'--',F6.0,1X,F6.2,1X,F6.4,1X,F5.2,3X,
     [illegible]
     [illegible]
 6004 FORMAT(   32X,'NO. OF     FOOTAGE     WEIGHT     SPEED'
     1,'     ROT. TIME     WEAR',/,33X,'BITS',5X,'PER BIT'
     2,24X,'PER BIT',5X,'T  B',//,34X,I2,6X,F6.1,5X,F5.1
     [illegible]
 6020 FORMAT(27X,'BIT NO.   INTERVAL     WEIGHT  SPEED    '
     1,'ROT. TIME  TRIP TIME  EQUIV. COST  WEAR',/,87X,
     2'(RIG HOURS)  T  B',/)
 6006 FORMAT(29X,I2,2X,F7.1,'--',F7.1,2X,F3.0,4X,F4.0,3X,
     [illegible]
 6007 FORMAT(39X,'FORMATION ROTATING TIME  = ',F7.2,
     1' RIG HOURS',//,
     1        39X,'FORMATION TRIP TIME      = ',F7.2,
     1' RIG HOURS',//,
     1        39X,'FORMATION RIG TIME       = ',F7.2,
```

```
     1' RIG HOURS',//,
     2        39X,'FORMATION COST                  ',[illegible]
     1' RIG HOURS',//,
     2        39X,'FORMATION BIT COUNT             ',3X,I2,//)
 6008 FORMAT(39X,'CUMULATIVE ROTATING TIME  = ',F7.2,
     1' RIG HOURS',//,
     1        39X,'CUMULATIVE TRIP TIME      = ',F7.2,
     1' RIG HOURS',//,
     1        39X,'CUMULATIVE RIG TIME       = ',F7.2,
     1' RIG HOURS',//,
     2        39X,'CUMULATIVE COST           = ',F7.2,
     1' RIG HOURS',//,
     2        39X,'CUMULATIVE BIT COUNT      = ',3X,I2,//)
C     THE FOLLOWING ARE THE INPUT VARIABLES
C      IW1 - NUMBER OF SEARCH POINTS PER UNIT OF WEIGHT
C      IR1 - NUMBER OF SEARCH POINTS PER UNIT OF SPEED
C      IOPT1 - ZERO FOR (1), OTHERWISE NON ZERO
C      IOPT4 - NUMBER OF ELEMENTS IN THE FIBONACCI SERIES
C      IOPT5 - ZERO IF (2), OTHERWISE NON ZERO
C      IOPT6 - IF NON ZERO COST OF EACH TRIAL IS PRINTED

      ICRIT=0
      [illegible]
      CTRIP=0.
      [illegible]
      CF[illegible]=0.
      [illegible]
      STROT=0.
      [illegible]
      [illegible]

C     FOR INPUT DETAILS SEE PROGRAM ON 'POINT' OPTIMIZATION
C     ADDITIONAL INPUT IS AS FOLLOWS
C      NOSET1 - IF NOT ZERO ANOTHER DATA SET EXISTS
C      NOSET2 - IF NOT ZERO ANOTHER DATA SET EXISTS AFTER [illegible]
C               SWITCHES HAVE BEEN READ
C

      READ(5,1) HOLE,BITCST,RIGCST,TRIPT,DEPTIN,P
      READ(5,3) SDEPTH,TDEPTH,AF,CF,DFS,FORM,FORMK
     [illegible]

C      W( ) - WEIGHT VALUES USED IN GRID SEARCH
C      R( ) = SPEED VALUES USED IN GRID SEARCH
C      F(I, ) = BEST FOOTAGE VALUE WITH I=1
C               TRIAL VALUE WITH I=2
C      SIMILARLY FOR [illegible](I, ), [illegible](I, ) AND TIME(I, )
C
```

```
C                  MAIN              ...(CONT'D)

      [illegible]
        BITHR=BITCST/RIGCST
       TRIPHR=TRIPT*1000.
      IF(FORM.GT.0.5)    IJK=2
      [illegible]
      XRANGE=XHI(1)-XLO(1)
      RRANGE=XHI(2)-XLO(2)
      [illegible]
      [illegible]
      [illegible]
      IF(IR.EQ.1) IR=2
      RANGE=TDEPTH-SDEPTH
      CALL TEST1(XLO(1),XLO(2),FOOTIN)
      FCOST=REPTIM+(SDEPTH+TDEPTH)*TRIPT/2.+BITHR
      MINBIT=IFIX(RANGE/FOOTIN)+1
      DO 12 J=1,IN
      X(J)=XLO(1)+XRANGE*FLOAT(J-1)/FLOAT(IN-1)
   12 CONTINUE
      [illegible]
        [illegible]+RRANGE*FLOAT(J-1)/FLOAT(IR-1)
   13 CONTINUE
      [illegible]
      DO 50 J=MINBIT,100
      [illegible]
      CALL SEARCH(FIN,WBEST,RBEST,TIN,DOUT,BOUT)
      TIM =FLOAT(J)*(TIN+FCOST)
      IF(TIM .GE.TOPT) GO TO 1000
      [illegible]
      WTEMP=WBEST
      RTEMP=RBEST
      TTEMP=TIN
      DTEMP=DOUT
      BTEMP=BOUT
      FTEMP=FIN
   50 CONTINUE
 1000 NBIT=J -1
      IF(IOPT1.NE.0) GO TO 2000
      [illegible]
      IIR=IFIX(8.*RTEMP+0.5)
      STROT=FLOAT(NBIT)*TTEMP
      STRIP=FLOAT(NBIT)*((TDEPTH+SDEPTH)*TRIPT/2.+REPTIM)
       SRIG=STROT+STRIP
      STEQ=SRIG+FLOAT(NBIT)*BITHR
      WRITE(6,6000)
      [illegible]
      WRITE(6,6002) SDEPTH,TDEPTH,AF,CF,DFS,RESP(IJK)
     [illegible]
     2TRIPHR,REPTIM,[illegible](IP)
      WRITE(6,6003)
      WRITE(6,6004) NBIT,FTEMP,WTEMP,RTEMP,TTEMP,IIR,IIS
```

```
C                    MAIN            ...(CONT'D)

      CTROT=CTROT+STROT
      CTRIP=CTRIP+STRIP
      CRIG=CRIG+SRIG
      CEQ=CEQ+STEQ
      ICBIT=ICBIT+NBIT
      WRITE(6,6007) STROT,STRIP,SRIG,STEQ,NBIT
      WRITE(6,6008) CTROT,CTRIP,CRIG,CEQ,ICBIT
      READ(5,2) NUSET1,NUSET2
      IF(NUSET1.NE.0) GO TO 99
      IF(NUSET2.NE.0) GO TO 999
      STOP
 2000 DO 2001 J=1,NBIT
      F(1,J)=FTEMP
      W(1,J)=WTEMP
      [illegible]
      [illegible]
      ID(1,J)=IFIX(8.*DTEMP+0.5)
      IR(1,J)=IFIX(8.*RTEMP+0.5)
 2001 CONTINUE
      IF(NBIT.EQ.1) GO TO 3000
      NBIT1=NBIT-1
      IF(IOPT5.NE.0) GO TO 700
      [illegible]
      AA3=0.
      RR1=1.E+50
      [illegible]
      [illegible]
      K=1.
   93 DO 90 J=1,NBIT
      F(2,J)=FTEMP+FLOAT(2*J-(NBIT+1))*A2
      CALL SEARCH(F(2,J),W(2,J),R(2,J),TIME(2,J),
     1DOUT,ROUT)
      ID(2,J)=IFIX(8.*DOUT+0.5)
      [illegible]
   90 CONTINUE
      FEET=SDEPTH
      COST=0.
      DO 91 J=1,NBIT1
      FEET=FEET+F(2,J)
   [illegible]
      COST=COST+TIME(2,NBIT)+(SDEPTH+TDEPTH)*TRIPT/2.
     1+FLOAT(NBIT)*(DEPTIM+BITPR)
      GO TO (105,500,92),K
  105 IF(IOPT5.NE.0) GO TO 710
      RR1=RR2
      [illegible]
      RR3=COST
      AA1=AA2
      AA2=AA3
      AA3=A2
```

```
C                     MAIN              ...(CONT'D)
      IF(RR2-RR3)500,104,104
[illegible] A2=A2+A2
      GO TO 93
500   IF(K.NE.1) GO TO 502
      [illegible]
      [illegible]
      DO 501 I=3,IOPT4
501   IFIB(I)=IFIB(I-1)+IFIB(I-2)
      FIB=FLOAT(IFIB(IOPT4))
      K=2
      JJ=1
[illegible] GO TO (590,591,592,593,594),JJ
[illegible] RANGE=AA3-AA1
      DEL=RANGE*FLOAT(IFIB(IOPT4-2))/FIB
      A2=AA1+DEL
      X1=A2
      JJ=2
      GO TO 93
591   FX1=COST
      A2=AA3-DEL
      Y1=A2
      JJ=3
      GO TO 93
      [illegible]
      [illegible]
       KKK=IOPT4-2
      DO 520 I=1,KKK
      [illegible]
      [illegible]
      [illegible]
      IF(FX1.GT.FY1) GO TO 523
      AA3=Y1
      Y1=X1
      FY1=FX1
      X1=AA1+DEL
      A2=X1
      JJ=4
      GO TO 93
      FX1=COST
      GO TO 520
[illegible] AA1=X1
      X1=Y1
      FX1=FY1
      [illegible]
      A2=Y1
      JJ=5
      GO TO 93
594   [illegible]
      CONTINUE
      IF(FX1.GT.FY1) GO TO 536
```

```
      C                    MAIN            ...(CONT'D)
      A2=X1
      K=3

  536 A2=Y1
      K=3
      GO TO 93
   92 DO 94 J=1,NBIT
      F(1,J)=F(2,J)
      [illegible]
      [illegible]
      TIME(1,J)=TIME(2,J)
      [illegible]
      [illegible]
   94 CONTINUE
      FEET=SDEPTH
      [illegible]
      [illegible]
      WRITE(6,6002) SDEPTH,TDEPTH,AF,CF,CFS,RESP(IJK),
     1FORMK,XLO(1),XHI(1),XLO(2),XHI(2),HOLE,NITEN,
     2TRIPHR,DEPTIN,PLAN(IP)
      [illegible]
      WRITE(6,6020)
      DO 3001 J=1,NBIT
      [illegible]
      FEET=FEET+F(1,J)
      [illegible]
      TEQ=TIME(1,J)+TRIP+RITCST/RIGCST
      [illegible]
     1TRIP,TEQ,IC(1,J),IB(1,J)
      STROT=STROT+TIME(1,J)
      STRIP=STRIP+TRIP

 3001 CONTINUE
      SRIG=STROT+STRIP
      CTROT=CTROT+STROT
      CTRIP=CTRIP+STRIP
      CRIG=CRIG+SRIG
      CEQ=CEQ+STEQ
      ICNBIT=ICNBIT+NBIT
      WRITE(6,6007) STROT,STRIP,SRIG,STEQ,NBIT
      WRITE(6,6008) CTROT,CTRIP,CRIG,CEQ,ICNBIT
      READ(5,2) NUSET1,NUSET2
      IF(NUSET1.NE.0) GO TO 49
      IF(NUSET2.NE.0) GO TO [illegible]
      STOP
  700 A2=0.
      K=1
      KK=1
       KJ=1
      ADD=FTEMP/1000.
```

```
      C                 AI        [illegible]
        KNT=0
        [illegible]
  705   A2=A2+ADD
  710   DELC=COST-TOPT
        [illegible]
  711   TOPT=COST
        AA2=A2
        KNT=0
  712   IF(IOPT6.EQ.0) GO TO 716
        IF(KK.LT.53) GO TO 715
        [illegible]
        KK=1
  715   WRITE(6,720) A2,COST,DELC
        [illegible]
  716   KJ=KJ+1
        KNT=KNT+1
        IF(KNT.GT.14) GO TO 750
        IF(KJ.GT.100) GO TO 750
        GO TO 705
  750   A2=AA2
        K=3
        GO TO 95
  720   FORMAT(10X,'A2  ',F4.1,10X,'COST  [illegible]
        [illegible]
        [illegible]
        END
```

SUBROUTINE SEARCH

```
       SUBROUTINE SEARCH(FOOTIN,XBEST,RBEST,TBEST,
      1DBEST,BBEST)
C
C     THIS SUBROUTINE CALCULATES THE BEST WEIGHT AND SPEED
C     BY A GRID SEARCH METHOD.  THE ASSOCIATED COST AND THE
C     FINAL WEAR CONDITIONS ARE ALSO FOUND.
C

       COMMON AF,CF,DFS,FORM,FORMR
       COMMON TDEPTH,SDEPTH,HOLE,RIGCST,BITCST,TRIPT,DEPTI
       COMMON NO1,NOMAX,IW,IR
       COMMON XLO(2),XHI(2),ERR,W(550),R(1400),P
      [illegible]
       ITEMP=IW
       DO 13 K=1,IR
       IND=0
       IJ=ITEMP
       IF(ITEMP.EQ.0) GO TO 14
       DO 12 JJ=1,IJ
       IF(IND.EQ.1) GO TO 13
      [illegible]
       CALL TEST(W(J),R(K),FOOTIN,FX,DX,BX)
       IF(FX.GT.1.E+09) GO TO 15
       IND=1
       IF(FX.GE.TBEST) GO TO 12
       TBEST=FX
       RBEST=R(K)
       XBEST=W(J)
       DBEST=DX
       BBEST=BX
      [illegible]
   [illegible] ITEMP=J-1
   [illegible] CONTINUE
   [illegible] CONTINUE
   [illegible] RETURN
       END
```

SUBROUTINE TEST1

```
      SUBROUTINE TEST1(WEIGHT,REVS,FOOTIN)
C     THIS SUBROUTINE CALCULATES THE FOOTAGE POSSIBLE USING A
C     GIVEN WEIGHT AND SPEED.
C
      COMMON AF,CF,DFS,FOOT,FOPK
      COMMON TDEPTH,SDEPTH,HOLE,BICCST,PIPCST,TRIPT,DEPTI
      COMMON R01,ROMAX,IN,IP
      COMMON XLO(2),XHI(2),EN,W(550),Y(1400),P
      DULLA(D)= 1.0-(6.0+.928125*D)*D
      DULLAI(D)=(1.0+(3.+.309375*D)*D)*D
      [illegible]
      SIN1(D)=(1.019770+(1.2923.4+(.1123279*D-.4137675)
     1*D)*D)*D
      IP=IFIX(R/0.9)+1
      IF(AF.LT.1.E-05) AF=1.
      A=EXP(-100./REVS**2)
      [illegible]
 1      ARR=A*REVS**0.75+0.5*REVS*(1.-A)
      GO TO 3
 2      ARR=A*REVS**0.429+0.2*REVS*(1.-A)
 3    WBAR=7.88*WEIGHT/HOLE
      ENYATE=1359.1-714.19*ALOG10(WBAR)
      ROTI=REVS*(1.+.0000434*REVS**2)
      [illegible]
      F1=AF*ENYATE/ROTI
      F2=F1*CF*(WBAR**ROOTK)*ARR
      F3=F1*REVS/DFS/YLOAD
      GO TO (20,20,30),IP
20    POUT=F3*DULLAI(1.)
      IF(POUT.LE.1.0) GO TO 10
      X=1.
 6    DMAX=X-(F3*DULLAI(X)-1.)/(F3*DULLA(X))
      IF(ABS((DMAX-X)/DMAX).LE.1.E-05) GO TO 5
      X=DMAX
      GO TO 6
10    DMAX=1.
5     GO TO (4,40,30),IP
 4    FOOTIN=F2*SIN1(DMAX)
      RETURN
40    FOOTIN=F2*DMAX
      RETURN
30    FOOTIN=F2/F3
      RETURN
      END
```

SUBROUTINE TEST

```
      SUBROUTINE TEST(WEIGHT,REVS,FOOTIN,TEFF,DOUT,ROUT)
C
C     THIS SUBROUTINE CALCULATES THE TIME TO DRILL A GIVEN
C     FOOTAGE USING THE GIVEN WEIGHT AND SPEED.  THE FINAL
C     WEAR CONDITIONS ARE ALSO CALCULATED.
C
      COMMON AF,CF,DFS,FORM,FORMK
      COMMON TDEPTH,SDEPTH,HOLE,RIGCST,BITCST,TRIPT,SEPTI
      COMMON W01,WMAX,IN,IP
      COMMON XLO(2),XHI(2),ERR,[illegible](550),[illegible](1400),P
       DULLA(D)= 1.0+(6.0+.328125*D)*D
      DULLAI(D)=(1.0+(3.+.109375*D)*D)*D
      [illegible]
      SINT(D)=[illegible]
     1*D)*D)*D
      IP=IFIX(P/0.9)-1
      A=EXP(-100./REVS**2)
      IF(FORM-0.5) 1,1,2
 1      ARR=A*REVS**0.75+0.5*REVS*(1.-A)
      GO TO 3
 2      ARR=A*REVS**0.428+0.2*REVS*(1.-A)
 3    WRAR=7.88*WEIGHT/HOLE
      [illegible]
      ROTI=REVS*(1.+.0000434*REVS**2)
      XLOAD=21338.87/(1.+.031197*WRAR)**3.2258
      GO TO (20,20,30),IP
20    F1=AF*EXRATE/ROTI
      F2=F1*CF*(WRAR**FORMK)*ARR
      F3=F1*REVS/DFS/XLOAD
      GO TO (40,50,30),IP
40    C=FOOTIN/F2
      [illegible]
 4    DOUT=X-(SINT(X)-C)/DULLA(X)
      IF(ABS((DOUT-X)/DOUT).LT.1.E-05) GO TO 5
      X=DOUT
      GO TO 4
 5    IF(DOUT.GT.1.) GO TO 10
      DDIFF=DULLAI(DOUT)
      [illegible]
      IF(ROUT.GT.1.) GO TO 10
      [illegible]
      GO TO 11
10    TEFF=1.E+10
11    RETURN
50    DOUT=FOOTIN/F2
      IF(DOUT.GT.1.0) GO TO 10
      DDIFF=DULLAI(DOUT)
      ROUT=F3*DDIFF
      IF(ROUT.GT.1.0) GO TO 10
```

SUBROUTINE TEST ...(CONT'D)

```
      TEFF=F1*DDIFF
      RETURN
[illegible]
      BOUT=FOOTIN/( F1*CF*(VBAR**EOX-K)*ANK)
      IF(BOUT.GT.1.0) GO TO 10
[illegible]
      RETURN
```

```
          C              MAIN

C            MAIN

C      THE PROGRAM USES THE DYNAMIC PROGRAMMING APPROACH
C      DESCRIBED IN SECTION III-D TO ALLOCATE THE DISTANCE
C      TO BE DRILLED BY EACH BIT IN AN OPTIMAL MANNER
C
C      IN MANY RESPECTS THIS PROGRAM IS SIMILAR TO THE SIMPLE
C      INTERVAL PROGRAM AND SO ONLY THE DIFFERENCES WILL BE
C      INDICATED.



       DIMENSION FINT(40),WINT(40),RINT(40),TROT(40),DINT
      1(40),BINT(40),TTRIP(40),TEQ(40),IL(40),IB(40),
      1RESP(2),WEAR(3),FOUT(41)
       COMMON AF,CF,DFS,FORM,FORMK,FITHR
         COMMON TDEPTH,SDEPTH,HOLE,RIGCST,BITCST,TRIPT,REPTIM
       COMMON NO1,NOMAX,IN,IK,P,IM,IDEG1,IDEG2
       COMMON XLO(2),XHI(2),ERR,RF,FIN(100),TIN(100),W(550)
       COMMON R(1400),TIME(40,50),F(40,50)
       DATA WEAR/'SS','FC','TC'/
       DATA RESP/'SOFT','HARD'/
     1 FORMAT(8F10.3)
     2 FORMAT(40I2)
     3 FORMAT(5F10.3,5F5.1)
  6000 FORMAT('1',///,52X,'SUMMARY OF INPUT DATA',/)
  6001 FORMAT(5X,'INTERVAL          AF      CF       S     SPEED',
      1'     WEIGHT            CONSTRAINTS          BIT SIZE  BIT'
      2,' COST   TRIP RATE  CHANGE TIME  WEAR',/,37X,
      3'RESPONSE RESPONSE  WMIN  WMAX  NMIN  NMAX  (INCHES)'
      4,'    (HOURS)    (HOURS)       (HOURS)      TYPE',/)
  6003 FORMAT(53X,'P R O C E D U R E',// )
  6002 FORMAT(2X,F6.0,'--',F5.0,1X,F6.2,1X,F6.4,1X,F5.2,5X,
      [illegible]
      [illegible]
  6020 FORMAT(27X,'BIT NO.   INTERVAL     WEIGHT  SPEED    '
      1,'ROT. TIME  TRIP TIME  EQUIV. COST  WEAR',/,37X,
      2'(RIG HOURS)  T   B',/)
  6006 FORMAT(29X,I2,2X,F7.1,'--',F7.1,2X,F5.0,4X,F4.0,5X,
      1F7.2,4X,F7.2,5X,F7.2,5X,I1,2X,I1,//)
  6007 FORMAT(39X,'FORMATION ROTATING TIME  = ',F7.2,
      1' RIG HOURS',//,
      1          39X,'FORMATION TRIP TIME      = ',F7.2,
      1' RIG HOURS',//,
      1          39X,'FORMATION RIG TIME       = ',F7.2,
      1' RIG HOURS',//,
      2          39X,'FORMATION COST           = ',F7.2,
      1' RIG HOURS',//,
      2          39X,'FORMATION BIT COUNT      = ',3X,I2,///)
  6008 FORMAT(39X,'CUMULATIVE ROTATING TIME = ',F7.2,
      1' RIG HOURS',//,
```

```
      C                 MAIN            ...(CONT'D)

     1        39X,'CUMULATIVE TRIP TIME       = ',F7.2,
     1' [illegible]',//,
     1        39X,'CUMULATIVE RIG TIME        = ',F7.2,
     1' RIG HOURS',//,
     2        39X,'CUMULATIVE COST            = ',F7.2,
     1' RIG HOURS',//,
     2        39X,'CUMULATIVE BIT COUNT       = ',3X,I2,//)
C     NEW INPUT VARIABLES
C
C      IDEG1 - DEGREE OF INTERPOLATION TO BE USED WHEN
C      IDEG2 - DEGREE OF INTERPOLATION USED WHEN FINDING COST
C      NO1 - THE NUMBER OF DIFFERENT FOOTAGES FOR WHICH A
C            ROTATING TIME IS CALCULATED.
C       NOMAX - THE NUMBER OF STATES AT EACH STAGE.
C        ENSET - NON ZERO VALUE RESULTS IN READING ANOTHER
C                DATA SET
C

       [illegible]
      ICNIT=0
      CTROT=0.
      [illegible]
      CRIG=0.

  999 READ(5,2) IN1,IR1,IDEG1,IDEG2,NO1,NOMAX
   99 [illegible]
      [illegible]
      STEQ=0.
      READ(5,1) HOLE,BITCST,RIGCST,TRIPT,REPTIM,P
      READ(5,3) SDEPTH,TDEPTH,AF,CF,DFS,FORM,FORMN,
     1(XLO(J),J=1,2),(XHI(J),J=1,2)

C     INTERNAL VARIABLES
C
C      FIN( ) - FOOTAGE VALUES FOR WHICH ROTATING TIMES ARE
C               FOUND
C      TIN( ) - ROTATING TIMES ASSOCIATED WITH FIN( )
C      F(J, ) - DEPTHS AT STAGE J
C      TIME(J, ) - VALUES OF OPTIMUM COSTS ASSOCIATED WITH
C                  F(J, )
C      WINT(J) - OPTIMUM WEIGHT FOR THE STAGE J
C      RINT(J) - OPTIMUM SPEED FOR THE STAGE J
C      TROT(J) - ROTATING TIME FOR STAGE J
C      TTRIP(J) - TRIP TIME ASSOCIATED WITH STAGE J
C      TEQ(J) - COST ASSOCIATED WITH STAGE J
C      FOUT(J) - DEPTH AT TOP OF STAGE J
C

      IP=IFIX(P/0.9)+1
```

```
C                    MAIN              ...(CONT'D)
      PITHRS=PITCST/RIGCST
     TRIPHR=TRIPT*1000.
[illegible]
[illegible]
    WRANGE=XHI(1)-XLO(1)
    RRANGE=XHI(2)-XLO(2)
     IW=IN1*IFIX(WRANGE+0.5)+1
    IR=IN1*IFIX(RRANGE+0.5)+1
    IF(IW.EQ.1) IW=2
    IF(IR.EQ.1) IR=2
    DO 16 J=1,IW
16  W(J)=XLO(1)+WRANGE*FLOAT(J-1)/FLOAT(IW-
    DO 17 J=1,IR
17  R(J)=XLO(2)+RRANGE*FLOAT(J-1)/FLOAT(IR-1)
    RANGE=TDEPTH-SDEPTH
    CALL TEST1(XLO(1),XLO(2),FOOTIN)
    FOOTIN=FOOTIN*.999
    IF(RANGE.GT.FOOTIN) FMAX=FOOTIN
    IF(RANGE.LE.FOOTIN) FMAX=RA[illegible]
    CALL TEST1(XHI(1),XHI(2),FOOTIN)
    IF(FOOTIN.GT.FMAX) FOOTIN=F[illegible]
    FMIN=FOOTIN/2.
[illegible]
    DO 10 J=1,NO1
    FIN(J)=FMIN+RF*FLOAT(J-1)/FLOAT(NO1-1)
10  CONTINUE
    DO 11 J=1,NO1
    CALL SEARCH(FIN(J),WBEST,RBEST,TIN(J),DOUT,ROUT)
11  CONTINUE
    MINBIT=IFIX(RANGE/FMAX)+1
    IF(MINBIT.GT.40)  GO TO 1111
    TOPT=1.E+20
    DO 50 J=1,40
[illegible]
[illegible]
    FEND=TDEPTH-FLOAT(J)*(FIN(1)+3.)
    RANGE1=FEND-FSTART
    DO 70 K=1,NOMAX
70  F(J,K)=FSTART+RANGE1*FLOAT(K-1)/FLOAT(NOMAX-1)
    IF(J.EQ.1) GO TO 71
    CALL FIND(J,1)
    IF(MINBIT.GT.J) GO TO 72
    IF(TIME(J,1).GE.TOPT) GO TO 1000
    TOPT=TIME(J,1)
72  DO 73 K=2,NOMAX
[illegible]
    GO TO 50
71  DO 74 K=1,NOMAX
[illegible]
74  TIME(1,K)=XPOL1(DELF)+PITHRS+DEPTIM+(F(1,K)+TDEPTH)
```

```
        C                    MAIN             ...(CONTD)

      1*TRIPT/2.
       IF( INPIT.EQ.1) TCPT=TIME(L,1)
   50  CONTINUE
       GO TO 1111
 1000  LPIT=J-1
       FLOW=SDEPTH
       FOUT(1)=SDEPTH
        DO 4000 K=1,LPIT
       KK=LPIT+1-K
       IF(KK.EQ.1) GO TO 3000
       CALL FINDI(LPIT,KK,FLOW,FINT(K))
       GO TO 5000
 3000  FINT(K)=TDEPTH-FLOW
 5000  CALL SEARCH(FINT(K),WINT(K),RINT(K),FROT(K),DINT(K)
      *,PINT(K))
       ID(K)=IFIX(2.*DINT(K)+0.5)
       [illegible]
       TTRIP(K)=(FLOW+FINT(K)/2.)*TRIPT+[illegible]
       [illegible]
       STROT=STROT+TROT(K)
       STRIP=STRIP+TTRIP(K)
       STEQ=STEQ+TEQ(K)
       [illegible]
       FOUT(K+1)=FOUT(K)+FINT(K)
 4000  CONTINUE
       CTROT=CTROT+STROT
       [illegible]
       CEQ=CEQ+STEQ
       TRIPT=TRIPT*100.
       [illegible]
       [illegible]
       [illegible]
      [illegible]
      [illegible](2),DMI(2),ROLL,TITLE,TRIPT,DEPTI
      [illegible]
       [illegible]
       [illegible]
       [illegible]
 6005  WRITE(6,6006) J,FOUT(J),FOUT(J+1),WINT(J),RINT(J)
      [illegible]
      1,TEQ(J),ID(J),IR(J)
       SRIC=STROT+STRIP
       WRITE(6,6007) STROT,STRIP,SRIC,STEQ,LPIT
       [illegible]
       CRIC=CTROT+CTRIP
       [illegible]
       GO TO 1113
 [illegible]
 1112  FORMAT('      ERROR      ERROR      ERROR      ',/,'      MAX NO
      * OF PITS EXC
```

```
[illegible])
READ(5,2) E,SET
IF (ESET.EQ.0) GO TO 99
 [illegible]
```

FUNCTION XNPOL1

```
      FUNCTION XNPOL1(DELF)
C     THIS FUNCTION SUBPROGRAM INTERPOLATES THE APPROPRIATE
C     ROTATING TIME FOR A GIVEN FOOTAGE USING LAGRANGIAN
C     INTERPOLATION.
C

      COMMON AF,CF,DFS,FORM,FORMK,PITHR
        COMMON TDEPTH,SDEPTH,ROLL,RIGCST,BITCST,TRIPT,DEPTI
      COMMON NC1,NCMAX,IN,IR,R,IP,IDEG1,IDEG
      COMMON XLO(2),XHI(2),ERR,RF,FIN(100),TIN(100),N(550)
      COMMON H(1400),TIME(40,50),F(40,50)
      IF((DELF.LT.FIN(1)).OR.(DELF.GT.FIN(NC1))) GO TO 3
      K=IDEG1+1
    1 IF(DELF.LE.FIN(K)) GO TO 2
      K=K+1
      IF(K.GT.NC1) GO TO 3
      GO TO 1
    2 [illegible]
      J2=K
      XNPOL1=0.
      DO 11 J=J1,J2
      [illegible]
      [illegible]
      IF(J.EQ.K) GO TO 12
      XL=XL*(DELF-FIN(K))/(FIN(J)-FIN(K))
   12 [illegible]
      XNPOL1=X[illegible]
   11 CONTINUE
      RETURN
      XNPOL1=1.E+10
      RETURN
      END
```

FUNCTION XNPOL2

```
      FUNCTION XNPOL2(FPREV,NSTAGE)
C
C     THE FUNCTION SUBPROGRAM INTERPOLATES THE COST
C     ASSOCIATED WITH THE GIVEN VALUE OF THE DEPTH.
C

      COMMON AF,CF,DFS,FORM,FORMK,RITHR
     1 COMMON TDEPTH,SDEPTH,HOLE,RIGCST,BITCST,TRIPT,REPTI
      COMMON ND1,NOMAX,IW,IR,P,IP,IDEG1,IDEG2
      COMMON XLO(2),XHI(2),ERR,RF,FIN(100),TIN(100),W(550)
      COMMON R(1400),TIME(40,50),F(40,50)
      [illegible]
      IF(FPREV.LT.F(J,1)) GO TO 3
      [illegible]
    1 IF (FPREV.LE.F(J,K)) GO TO 2
      K=K+1
      IF(K.GT.NOMAX) GO TO 3
      GO TO 1
    2 [illegible]
      J2=K
      XNPOL2=0.
      DO 11 K=J1,J2
      XL=1.
      DO 12 L=J1,J2
      IF(L.EQ.K) GO TO 12
      [illegible]
   12 CONTINUE
      XNPOL2=TIME(J,K)*XL+XNPOL2
   11 CONTINUE
      RETURN
    3 XNPOL2=1.E+10
      RETURN
      END
```

SUBROUTINE FIND

```
      SUBROUTINE FIND(NSTAGE,NO)

     THIS SUBROUTINE CALCULATES THE OPTIMUM COST FOR A
     PARTICULAR VALUE OF THE STATE VARIABLE (F(NSTAGE,NO)).


      COMMON AF,CF,DFS,FORM,FORMK,BITHR
        COMMON TDEPTH,SDEPTH,HOLE,RIGCST,BITCST,TRIPT,REPTI
      COMMON NO1,NOMAX,IW,IR,P,IP,IDEG1,IDEG2
      COMMON XLO(2),XHI(2),ERR,RF,FIN(100),TIN(100),W(550)
      COMMON R(1400),TIME(40,50),F(40,50
      [illegible]
      TIME(NSTAGE,NO)=1.E+10
      DELF=(TDEPTH-F(NSTAGE,NO))/FLOAT(NSTAGE)-1.
      DELF1=DELF
  999 X=XNPOL1(DELF)+(F(NSTAGE,NO)+DELF/2.)*TRIPT+BITHR+
     1XNPOL2((F(NSTAGE,NO)+DELF),NSTAGE)+REPTI
      [illegible]
      [illegible]
      IF(ABS(SS).LT.ERR) RETURN
      [illegible]
      GO TO 999
    1 TIME(NSTAGE,NO)=X
      [illegible]
      SS=SS*2.
      [illegible]
      GO TO 999
      END
```

SUBROUTINE FIND1

```
      SUBROUTINE FIND1(LBIT,NSTAGE,FNOW,DELF1)
C     THE SUBROUTINE CALCULATES THE OPTIMUM DISTANCE TO BE
C     DRILLED AT A PARTICULAR STAGE IN THE PROCESS.
C
      COMMON AF,CF,DFS,FORM,FRANK,BITHR
        COMMON TDEPTH,SDEPTH,HOLE,RIGCST,BITCST,TRIPT,REPTI
      COMMON NO1,NMAX,IN,IR,P,IP,IDEG1,IDEG2
      COMMON XLO(2),XHI(2),EPR,RP,FIN(100),TIN(100),W(350)
      COMMON R(1400),TIME(40,50),F(40,50)
      SS=-3.
      TIM=1.E+10
      DELF=(TDEPTH-SDEPTH)/FLOAT(LBIT)    -1.
      DELF1=DELF
  999 X=XNPOL1(DELF)+(FNOW+DELF/2.)*TRIPT+BITHR+XNPOL2
     1 ((FNOW+DELF),NSTAGE)+REPTI
      IF(TIM.GT.X) GO TO 1
      [illegible]
      [illegible]
      DELF=DELF1+SS
      GO TO 999
    1 TIM=X
      SS=SS*2.
      [illegible]
      DELF=DELF1+SS
      GO TO 999
      END
```

SUBROUTINE SEARCH

```
 SUBROUTINE SEARCH(FOOTIN,WBEST,RBEST,TBEST,
1DBEST, BBEST)
 COMMON AF,CF,DFS,FORM,FORMN,PITCH
   COMMON TDEPTH,SDEPTH,HOLES,AICOST,PITCST,TWIDTH,REPTI
 COMMON NO1,NOMAX,IW,IR,H,IT,IDEG1,IDEG2
 COMMON XLO(2),XHI(2),ERR,PF,FIN(100),TIN(100),W(350)
 COMMON R(1400),TIME(40,50),F(40,50)
 TBEST=1.E+10
 ITEMP=IW
 DO 13 K=1,IR
 IND=0
 IJ=ITEMP
 IF(ITEMP.EQ.0) GO TO 14

 IF(IND.EQ.1) GO TO 13

 CALL TEST(W(J),R(K),FOOTIN,FX,DX,BX)
 IF(FX.GT.1.E+09) GO TO 15
 IND=1
 IF(FX.GE.TBEST) GO TO 12
 TBEST=FX
 RBEST=R(K
 WBEST=W(J)
 DBEST=DX
 BBEST=BX
 GO TO 12

 CONTINUE
 CONTINUE
 RETURN
 END
```

SUBROUTINE TEST1

```
      SUBROUTINE TEST1(WEIGHT,REVS,FOOTIN)
      COMMON AF,CF,DFS,FORM,FORMN,WITHN
       COMMON TDEPTH,SDEPTH,HOLE,RIGCST,BITCST,TRIPT,ALPTIM
      COMMON NC1,WMAX,IW,IN,P,IP,IDEG1,IDEG2
      COMMON XLO(2),XHI(2),EXN,RP,FIN(200),TIN(200),W(550)
      COMMON R(100),TIME(40,50),F(40,50)
       DULLA(D)= 1.0+(6.0+.922125*D)*D
      DULLAI(D)=(1.0+(3.+.307375*D)*D)*D
      DULLAM(D)=(1.0+(6.0+.328125*D)*D)**0.5
      SIN1(D)=(1.019775+(1.232304+(.113227*D-.4127675)
     1*D)*D)*D
      IP=IFIX(P/0.9)+1
      IF(AF.LT.1.E-05) AF=1.
            [illegible]
      IF(FORM-0.9) 1,1,2
    1       [illegible]
          [illegible]
    2   ARK=A*REVS**0.425+0.2*REVS*(1.-A)
    3 WBAR=7.88*WEIGHT/HOLE
                [illegible]
        [illegible]
      XLOAD=21335.57/(1.+.0311*WBAR)**3.225
      F1=AF*EXRATE/ROTI
      F2=F1*CF*(WBAR**FORMN)*ARK
      F3=F1*REVS/DFS/XLOAD
      GO TO (20,20,30),IP
      DOUT=F3*DULLAI(1.)
      IF(DOUT.LE.1.0) GO TO 10
      X=1.
      DMAX=X-(F3*DULLAI(X)-1.)/(F3*DULLA(X))
      IF(ABS((DMAX-X)/DMAX).LE.1.E-05) GO TO 60
      X=DMAX
      GO TO 6
      DMAX=1.
      GO TO (4,40,30),IP
      FOOTIN=F2*SIN1(DMAX)
      RETURN
      FOOTIN=F2*DMAX
      RETURN
      FOOTIN=F2/F3
      RETURN
      END
```

SUBROUTINE TEST

```
      SUBROUTINE TEST(HEIGHT,REVS,FOOTIN,TEFF,DOUT,ROUT)
      COMMON AF,CF,DFS,FORM,FORMK,WITHR
        COMMON TDEPTH,SDEPTH,HOLE,PICCST,BITCST,TRIPT,RENTI
      COMMON DO1,LDAY,IW,IF,P,IP,INDS1,INDS2
      COMMON XLO(2),XHI(2),ERR,PH,FIN(100),TIN(100),H(500)
      COMMON W(1400),TIME(40,50),F(40,50)
       DULLA(D)= 1.0+(5.0+.928125*D)*D
      DULLAI(D)=(1.0+(3.+.309375*D)*D)*D
      [illegible]
      [illegible]
     1*D)*D)*D
      IP=IFIX(P/ .9)+1
      [illegible]
      IF(FORM-0.5) 1,1,2
    1   AFP=A*REVS**0.75+0.5*REVS*(1.-A)
      GO TO 3
    2   AFP=A*REVS**0.428+0.2*REVS*(1.-A)
    3 WMAC=7.86*HEIGHT/HOLE
      [illegible]
      [illegible]
      [illegible]
      GO TO (20,20,30),IP
      [illegible]
      F2=F1*CF*(LDAY**FORMK)*AFP
      F3=F1*REVS/DFS/XLOAD
      GO TO (40,50,30),IP
      [illegible]
      X=.0
      [illegible]
      IF(ABS((DOUT-X)/DOUT).LT.1.E-03) GO TO 5
      X=DOUT
      GO TO 4
    5 IF(DOUT.GT.1.) GO TO 10
      DDIFF=DULLAI(DOUT)
      ROUT=F3*DDIFF
      IF(ROUT.GT.1.) GO TO 10
      [illegible]
      GO TO 11
   10 TEFF=1.E+10
   11 RETURN
   20 DOUT=FOOTIN/F2
      IF(DOUT.GT.1.0) GO TO 10
      DDIFF=DULLAI(DOUT)
      ROUT=F3*DDIFF
      [illegible]
      TEFF=F1*DDIFF
      RETURN
   30 F1=DFS*XLOAD/REVS
      DOUT=FOOTIN/(F1*CF*(LDAY**FORMK)*AFP)
      IF(DOUT.GT.1.0) GO TO 10
```

```
RESULTS FOR BASIC INTERVAL METHOD - AFTER THE BITS
DRILL EQUAL PARTS OF THE INTERVAL.  THERE [illegible]
GROUP FOR EVERY CHANGE IN THE CONSTANTS.

                    GROUP NO.  1

     BIT NO.       INTERVAL     WEIGHT   SPEED

        1       1400. -  1907.   59.      133.
        2       1907. -  2413.   59.      133.
        3       2413. -  2920.   59.      133.

                    GROUP NO.  2

     BIT NO.       INTERVAL     WEIGHT   SPEED

        1       2920. -  3345.   59.      128.
        2       3345. -  3770.   59.      128.

                    GROUP NO.  3

     BIT NO.       INTERVAL     WEIGHT   SPEED

        1       3770. -  4095.   59.      130.
        2       4095. -  4420.   59.      130.
        3       4420. -  4745.   59.      130.
        4       4745. -  5070.   59.      130.

                    GROUP NO.  4

     BIT NO.       INTERVAL     WEIGHT   SPEED

        1       5070. -  5495.   59.      128.
        2       5495. -  5920.   59.      128.
        3       5920. -  6345.   59.      128.
        4       6345. -  6770.   59.      128.

                    GROUP NO.  5

     BIT NO.       INTERVAL     WEIGHT   SPEED

        1       6770. -  6810.   59.       60.

                    GROUP NO.  6

     BIT NO.       INTERVAL     WEIGHT   SPEED

        1       6810. -  7145.   59.      108.
        2       7145. -  7480.   59.      108.
        3       7480. -  7815.   59.      108.
        4       7815. -  8150.   59.      108.
        5       8150. -  8485.   59.      108.
        6       8485. -  8820.   59.      108.
```

```
                GROUP NO.  7

BIT NO.      INTERVAL    WEIGHT  SPEED

   1     8520. - 8860.  50.      60.

                GROUP NO.  8

BIT NO.      INTERVAL    WEIGHT  SPEED

   1     8860. - 9020.  60.     100.

                GROUP NO.  9

BIT NO.      INTERVAL    WEIGHT  SPEED

   1     9020. - 9060.  50.      50.

                GROUP NO. 10

BIT NO.      INTERVAL    WEIGHT  SPEED

   1     9060. - 9270.  50.      50.

                GROUP NO. 11

BIT NO.      INTERVAL    WEIGHT  SPEED

   1     9270. - 10080.  52.     59.

                GROUP NO. 12

BIT NO.      INTERVAL    WEIGHT  SPEED

   1    10080. - 10415.  45.     65.
   2    10415. - 10750.  45.     65.

                GROUP NO. 13

BIT NO.      INTERVAL    WEIGHT  SPEED

   1    10750. - 10863.  45.     44.

   3    10975. - 11080.  45.     44.
   4    11080. - 11200.  45.     44.
   5    11200. - 11315.  45.     44.

                GROUP NO. 14

BIT NO.      INTERVAL    WEIGHT  SPEED

   1    11315. - 11650.  45.     70.
```

```
                      GROUP NO. 15

 BIT NO.        INTERVAL        WEIGHT   SPEED

              11???. - 11872.    45.
              11872. - 12014.    45.

                      GROUP NO. 16

 BIT NO.        INTERVAL        WEIGHT   SPEED

              12014. - 12363.    45.       55.
    2         12363. - 12641.    45.       55.
              12641. - 12920.    45.       55.

                      GROUP NO. 17

 BIT NO.        INTERVAL        WEIGHT   SPEED

              12920. - 13063.    45.
              13063. - 13216.    45.

                      GROUP NO. 18

 BIT NO.        INTERVAL        WEIGHT   SPEED

              13216. - 13363.    45.       70.
              13363. - 13510.    45.       70.

CUMULATIVE ROTATING TIME =  808.66  RIG HOURS
CUMULATIVE TRIP TIME     =  272.74  RIG HOURS
CUMULATIVE RIG TIME      = 1081.40  RIG HOURS
CUMULATIVE COST          = 1303.44  RIG HOURS
CUMULATIVE BIT COUNT     =    42
```

```
RESULTS FOR THE EMPIRICAL INVESTIGATION OF THE
'INTERVAL' OPTIMIZATION PROCEDURE.

                GROUP NO.  1

               INTERVAL   WEIGHT

             1400. -  1900.  60.
             1900. -  2406.  59.
             2406. -  2920.  59.

                GROUP NO.  2

               INTERVAL   WEIGHT

             2920. -  3345.  59.
             3345. -  3770.  59

                GROUP NO.  3

               INTERVAL   WEIGHT  SPEED

     1       3770. -  4094.  59.     100.
     2       4094. -  4419.  57.     120.
     3       4419. -  4744.  57.     100.
     4       4744. -  5070.  59.     125.

                GROUP NO.  4

PIT NO.        INTERVAL   WEIGHT

             5070. -  5484.  59.
     2       5484. -  5905.  59.
     3       5905. -  6334.  59.
     4       6334. -  6770.  59.

                GROUP NO.  5

PIT NO.        INTERVAL   WEIGHT

             6770. -  6810.  50.

                GROUP NO.  6

               INTERVAL   WEIGHT  SPEED

     1       6810. -  7139.  60.     109.
     2       7139. -  7469.  59.     110.
     3       7469. -  7802.  59.
     4       7802. -  8135.  59.     109.
     5       8135. -  8470.  59.     110.
     6       8470. -  8827.  59.     109.
```

GROUP [illegible]

PIT NO. INTERVAL [illegible]

[illegible] [illegible] 50.
[illegible] [illegible] 10 [illegible]

GROUP NO.

INTERVAL [illegible]

[illegible]

GROUP [illegible]

INTERVAL [illegible]

[illegible]

GROUP NO. 1[illegible]

INTERVAL HEIGHT [illegible]

[illegible]

PIT NO. INTERVAL [illegible]

1[illegible]
1[illegible]41[illegible]

GROUP [illegible]

[illegible] INTERVAL HEIGHT [illegible]

1 1[illegible]
2 10[illegible]62.
3 10[illegible]75. [illegible] 4[illegible]. [illegible]4.
4 [illegible] [illegible]5. [illegible]
5 [illegible] [illegible]. [illegible]4.

[illegible] 1[illegible]

[illegible] INTERVAL HEIGHT SPEED

[illegible] 110[illegible]. [illegible]. 70.

```
                GROUP NO. 15

   BIT NO.      INTERVAL     WEIGHT  SPEED

      2    11[illegible]2. - 12084.  45.     7[illegible].

                GROUP NO. 16

   BIT NO.      INTERVAL     WEIGHT  SPEED

      1    12084. - 12362.  45.     6[illegible].
      2    12362. - 12641.  45.     6[illegible].
      3    12641. - 1292[illegible].  45.     66.

                GROUP NO. 17

   BIT NO.      INTERVAL     WEIGHT  SPEED

      1    1292[illegible]. - 13067.  45.     [illegible].
      2    13067. - 13214.  45.     59.

                GROUP NO. 18

   BIT NO.      INTERVAL     WEIGHT  SPEED

      1    13214. - 13363.  45.     70.
           13363. - 135[illegible].  45.     7[illegible].

CUMULATIVE ROTATING TIME  = [illegible].37 RIG HOURS
CUMULATIVE TRIP TIME      =  2[illegible]2.0[illegible] RIG HOURS
CUMULATIVE RIG TIME       = 1[illegible]91.0[illegible] RIG HOURS
CUMULATIVE COST           = 1303.1[illegible] RIG HOURS
CUMULATIVE BIT COUNT      =     42
```

```
RESULTS FOR THE DYNAMIC PROGRAMMING VARIATION OF
THE 'INTERVAL' OPTIMIZATION

                 GROUP NO.  1

  UNIT NO.       INTERVAL      WEIGHT   SPEED

     1        1400. - 1934.     60.     120.
     2        1934. - 2399.     60.     122.
     3        2399. - 2920.     59.     125.

                 GROUP NO.  2

  UNIT NO.       INTERVAL      WEIGHT   SPEED

     1        2920. - 3342.     57.     125.
     2        3342. - 3770.     59.     127.

                 GROUP NO.  3

  UNIT NO.       INTERVAL      WEIGHT   SPEED

     1        3770. - 4077.
     2        4077. - 4410.     [illegible]     [illegible]
     3        4410. - 4731.     57.     121.
     4        4731. - 5373.

                 GROUP NO.  4

  UNIT NO.       INTERVAL      WEIGHT   SPEED

     1        5373. - 5475.     60.     131.
     2        5475. - 5915.     57.     124.
     3        5915. - 6357.     59.     126.
     4        6357. - 6770.     55.     126.

                 GROUP NO.  5

  UNIT NO.       INTERVAL      WEIGHT   SPEED

              6770. - 6910.     56.      60.

                 GROUP NO.  6

  UNIT NO.       INTERVAL      WEIGHT   SPEED

     1        6910. - 7142.     59.     109.
     2        7142. - 7475.     59.     109.
     3        7475. - 7809.     59.     109.
     4        7809. - 8143.     60.     106.
     5        8143. - 8480.     59.     108.
     6        8480. - 8820.     57.     107.
```

```
                 GROUP NO. 7

PIT NO.        INTERVAL    WEIGHT   SHEET

            [illegible]2. - [illegible]. 50.

                 GROUP NO. 8

               INTERVAL    WEIGHT

            [illegible]. - [illegible]. [illegible].

                 GROUP NO. 9

               INTERVAL    WEIGHT

            [illegible]. - [illegible]. [illegible].

                 GROUP NO. 10

               INTERVAL    WEIGHT

            [illegible]. - [illegible]27. [illegible].

                 GROUP NO. 11

               INTERVAL    WEIGHT

            [illegible] - [illegible]. 52.

                 GROUP NO. 12

               INTERVAL    WEIGHT

   1        [illegible]. - 104.5. 45.
            104.5. - 1075. 45.

                 GROUP NO. 13

PIT NO.        INTERVAL    WEIGHT

   1        1075[illegible]. - 10863. 45.
   2        10863. - 10976. 45.
   3        10976. - 110[illegible]. 45.
   4        11[illegible]. - 112[illegible]. 45.
            [illegible] - [illegible]. 45.

                 GROUP NO. 14

PIT NO.        INTERVAL    WEIGHT

            [illegible] - [illegible]. 45.
```

```
                    GROUP NO. 15

 BIT NO.        INTERVAL      WEIGHT   SPEED


    2      11[illegible]1. - 12034.   45.        70.

                    GROUP NO. 16

 BIT NO.        INTERVAL      WEIGHT   SPEED

    1      12034. - 12362.   45.        [illegible].
    2      12362. - 12640.   45.        [illegible].
    3      12640. - 12920.   45.        67.

                    GROUP NO. 17

 BIT NO.        INTERVAL      WEIGHT   SPEED

    1      12920. - 130[illegible].   45.        59.
    2      130[illegible]. - 13216.   45.        59.
                    GROUP NO. 18

 BIT NO.        INTERVAL      WEIGHT   SPEED

    1      13216. - 13562.   45.        7[illegible].
    2      13562. - 1[illegible]21 .   45.        7[illegible].

CUMULATIVE ROTATING TIME =   [illegible]  RIG HOURS
CUMULATIVE TRIP TIME     =   292.67 RIG HOURS
CUMULATIVE RIG TIME      = 1[illegible]21.49 RIG HOURS
CUMULATIVE COST          = 15[illegible].51 RIG HOURS
CUMULATIVE BIT COUNT     =     42
```

Appendix H

Program and Results of "Multi-interval" Optimization

```
           C                   MAIN

C                 MAIN

C     THE PROGRAM IS DESIGNED TO OPTIMIZE THE DRILLING OF
C     A WELL USING THE DYNAMIC PROGRAMMING METHOD OF SECTION
C     III-E.
C

      DIMENSION XLO(2),XHI(2),RESP(2),NAPINC(10) , WEAR(3)
      COMMON AF(150),CF(150),DFS(150)
      COMMON FORM(150),FORMK(150),DEPTH(151)
      COMMON XLIMLO(150,2),XLIMHI(150,2), DS1(151)
      COMMON WIN(150),FIN(150),TOTALT(151,15,15), DS2(151)
      COMMON XNDIC(15,15),N(15,15),R(15,15),DE(15),DS(15)
      COMMON ERR,AINC1,AINC2,DEPTIN,CONS(2),TRIPT ,W
      COMMON HOLE,BITCST,RIGCST,ROTTIM,COST,HOUT,LAG,DELT
      COMMON IP,XBAR
      COMMON IOPT3,IOPT4,IOPT5,IOPT6
      DATA RESP/'SOFT','HARD'/
      DATA WEAR/'SS','FC','TC'/
    1 FORMAT(8F10.3)
    2 FORMAT(40I2)
    3 FORMAT(5F10.3,6F5.1)
 6004 FORMAT('1',44X,'P O L I C Y',///)
 6005 FORMAT(16X,'SECTION   INTERVAL  WEIGHT  SPEED  ROT.'
     1,' TIME  TRIP TIME  COST     BA-',/,72X,'(HOURS) T  R'
     [illegible]
 6006 FORMAT(18X,I2,3X,F6.0,'-',F6.0,1X,F4.1,3X,F4.1,2X,
     1F7.2,22X,I1,2X,I1,///)
 6001 FORMAT(15X,'SUMMARY',1X,F6.0,'-',F6.0,15X,F7.2,4X,
     [illegible]
 6002 FORMAT(37X,'BIT CHANGED AT ',F6.0,' FEET',// )
 6000 FORMAT('1',55X,'SUMMARY OF INPUT DATA',///)
 6008 FORMAT(3X,'INTERVAL          AF     CF        S     SPEED',
     14X,'WEIGHT        CONSTRAINTS        BIT SIZE  BIT COST'
     1,'   TRIP RATE'
     2,2X,'CHANGE TIME YS XS D     BA-',/,35X,'RESPONSE',1X,
     3'RESPONSE W IN WMAX RMIN RMAX   (INCHES)     (HOURS)'
     4,3X,'(HOURS)        (HOURS)                    TYPE',/)
 6009 FORMAT(1X,F6.0,'-',F6.0,1X,F6.2,1X,F6.4,1X,F5.2,
     [illegible]
     2F6.3,4X,F6.3,2X,F4.2,'/1000FT',4X,F5.2,4X,I2,1X,I2,
     [illegible]
 6003 FORMAT(28X,'CUMULATIVE ROTATING TIME = ', F7.2,
     1' RIG HOURS',//,
     1          28X,'CUMULATIVE TRIP TIME     = ', F7.2,
     1' RIG HOURS',//,
     1          28X,'CUMULATIVE RIG TIME      = ', F7.2,
     1' RIG HOURS',//,
     2          28X,'CUMULATIVE COST          = ', F7.2,
     1' RIG HOURS',//,
```

```
    4          28X,'CUMULATIVE BIT COUNT        = ', I3,///)
6010 FORMAT(28X,'GROUP ROTATING TIME          = ', F7.2,
    1' RIG HOURS',//,
    1          28X,'GROUP TRIP TIME             = ', F7.2,
    1' RIG HOURS',//,
    2          28X,'GROUP RIG TIME              = ', F7.2,
    1' RIG HOURS',//,
    3          28X,'GROUP COST                  = ', F7.2,
    1' RIG HOURS',//,
    4          28X,'GROUP BIT COUNT             = ', I3,///)
6020 FORMAT('1',3X,'*******************************'
    1,/,4X,'*',10X,'WARNING',11X,'*',/,4X,'*',29X,'*',/,
    24X,'* DEPTHS ARE NOT IN SEQUENCE *',/,4X,
    3          '*******************************')
6021 FORMAT('1',3X,'*******************************',/,
    [illegible]
    2'* NO OF INTERVALS EXCEEDS 150 *',/,4X,'*',29X,'*'
    [illegible]
    4          '*******************************')

C            OUTPUT IS SIMILAR IN MANY RESPECTS TO THE PREVIOUS
C        PROGRAMS, AND SO ONLY THE DIFFERENCES WILL BE NOTED
C
C       INPUT DATA
C
C        LAG - DEGREE OF INTERPOLATION IN SUBROUTINE INPOL
C        IOPT1 - IF NON ZERO THE RESULTS OF THE 'BACKWARDS'
C                CALCULATIONS ARE PRINTED OUT.
C        IOPT2 - IF NON ZERO A MAP OF THE SEARCH AREA IS GIVEN
C        IOPT3 - FIRST DULLNESS ELEMENT FOR WHICH A MAP IS
C                GIVEN
C        IOPT4 - NUMBER OF DULLNESS ELEMENTS SEPARATING EACH
C                MAP EVALUATION.
C       IOPT5 - NUMBER OF THE FIRST BEARING WEAR ELEMENT FOR
C               WHICH A MAP IS PRODUCED.
C        IOPT6 - NUMBER OF BEARING WEAR ELEMENTS SEPARATING
C                EACH MAP EVALUATION.
C        MAPNO - THE NUMBER OF DISTINCT STAGES FOR WHICH A MAP
C                  IS PRODUCED.
C        MAPING - VECTOR WHOSE ELEMENTS GIVE THE STAGES AT
C                  WHICH A MAP WILL BE PRODUCED.
C
C
C        INO - THE NUMBER OF STAGES FOR THE FORMATION
C        INOS1 - THE NUMBER OF DULLNESS VALUES AT EACH STAGE.
C        INOS2 - THE NUMBER OF BEARINGS VALUES AT EACH STAGE.
C        IND1 - IF ZERO THERE IS ANOTHER FORMATION IN THE GROUP
C        IND2 - IF NON ZERO THE STATE VARIABLE MATRIX VALUES.
C        IND3 - IF NON ZERO THE CUMULATIVE COST FIGURES ARE NOT
C               PRINTED
```

```
C                   MAIN           ...(CONT'D)


      CHG=0.
      [illegible]
      CTRIP=0.
      KOUNT=0
 9999 READ(5,2) LAG,IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,
     1NARNO,(NARING(J),J=1,NARNO)
      READ(5,1) ERR,AINC1,AINC2,(CONV(J),J=1,2)
  999 ILEV=0
      SROT=0
      SEG=0.
      STRIP=0.
      READ(5,1) HOLE,BITCST,RIGCST,TRIPT,REPTI[illegible],P
   99 READ(5,2) INO,INOS1,INOS2,IND1,IND2,IND3
      [illegible]
      [illegible]
      IF(ILEV.LE.150) GO TO 98
      WRITE(6,6021)
      STOP
   98 READ(5,3) SDEPTH,TDEPTH,AAF,ACF,ADFS,AFORM,AFORK,
     1(XLO(J),J=1,2),(XHI(J),J=1,2)
      IF((ILEV1.NE.1).AND.(ABS(TEMP-SDEPTH).GT.0.5))
     1WRITE(6,6020)
      FOOT=(TDEPTH-SDEPTH)/FLOAT(INO)
      DO 10 J=ILEV1,ILEV
      DEPTH(J)=SDEPTH+FLOAT(IFIX(FLOAT(J-ILEV1)*FOOT+0.5))
      [illegible]
      CF(J)=ACF
      DFS(J)=ADFS
      [illegible]
      XLIMLO(J,2)=XLO(2)
      [illegible]
      XLIMHI(J,2)=XHI(2)
      FORM(J)=AFORM
      FORDK(J)=AFORK
      NOS1(J)=INOS1
      NOS2(J)=INOS2
   10 CONTINUE
      TEMP=TDEPTH
      IF(IND2.EQ.0) GO [illegible]

C     IF IND2 IS NOT ZERO THE SIZES OF THE STATE VARIABLE
C     GRID ARE READ IN HERE
C

      READ(5,2) (NOS1(J),J=ILEV1,ILEV),(NOS2(J),J=ILEV1,
     1[illegible]
   20 IF(IND1.EQ.0) GO TO 99
      [illegible]
```

```
C                    MAIN                 ...(C
      IL[illegible]
      NOS1(ILEV)=9
      NOS2(ILEV)=9
      DEPTH(ILEV)=TDS-T[illegible]
      DO 38 J=1,9
      DO 37 K=1,9
      [illegible]
      CONTINUE
      CONTINUE
      BITHR=BITCST/RIGCST
      TRIPHR=1000.*TRIPT
      KOUNT=1
      DO 121 KK=1,ILEV-1
      IF(KOUNT.NE.1) GO TO 31
      WRITE(6,5
      [illegible]
      KOUNT=2
   31 IF(KOUNT.LT.61) GO TO 33
      KOUNT=1
      GO TO 32
      KOUNT=KOUNT+2
      IP=IFIX(P/0.9)+1
      XHAF=7.08/HOLE
      IF(FORM(KK).GT.0.5) IJK=2
      [illegible]
      WRITE(6,6009) DEPTH(KK),DEPTH( [illegible] ,AF(KK),CF(KK),
     1DFS(KK),RECP(IJK),FORM(KK),ALI [illegible]
     2[illegible]
      [illegible]
  121 CONTINUE
      DCUT=0.
      DO 120 K=1,ILEV-1
      [illegible]
      KOUNT=KOUNT+1
      FOOTIN=DEPTH(JJ+1)-DEPTH(JJ)
      KL=NOS1[illegible]
      IF(P.GT.1.0) GO TO 1[illegible]
      XNOS=FLOAT(NOS1(JJ)-1)
      DO 15 J=1,KL
      [illegible]
      DF(1)=0.
      IF(NOS2(JJ).EQ.1) GO TO 17
      KM=NOS2(JJ)
      XNOS=FLOAT(NOS2(JJ)-1)
      DO 16 J=1,KM
   16 [illegible]
   17 DR(1)=0.
      IF(JJ.EQ.1) GO TO 120
      [illegible]
      IF(IOPT1.NE.0) CALL RESULT(JJ)
```

```
      C                   MAIN           ...(CONT'D)

      IF(IOPT2.EQ.0) GO TO 120
      DO 150 [illegible]=1,[illegible]
  150 IF(JJ.EQ.MAPINS([illegible])) CALL [illegible](JJ
  120 CONTINUE
      CALL TEST2(ALIMLO(1,1),ALIMLO(1,2),TEFF,1,1,IANS,1)
      CALL FIND(ALIMLO(1,1),ALIMLO(1,2),1,3,1,TEFF)
      DI(1)=R(1,1)

      KOUNT=0
      DO 130 JJ=1,ILEVM1
      ROUTIN=DEPTH(JJ+1)-DEPTH(JJ)
      IF(JJ.EQ.1) GO TO 8
      CALL TEST2(XLIMLO(JJ,1),XLIMLO(JJ,2),TEFF,1,1,IANS,JJ)
      IF(IANS.EQ.1) GO TO 11

      REPLAC=TIMEO+TRIPT*DEPTH(JJ)+DEPTIN+BITCST/RIGCST
      IF(REPLAC.LT.TOTALT(JJ,1,1)) GO TO 11
      CALL TEST2(R(1,1),R(1,1),TEFF,1,1,IANS,JJ)
       TOTRT=TOTRT+DELT
      IDE=IFIX(DOUT*[illegible].+0.5)
      IPR=IFIX(DOUT*[illegible].+0.5)
       NSEC=NSEC+1
      WRITE(6,6006) NSEC, DEPTH(JJ),DEPTH(JJ-1),R(1,1),
     1R(1,1),DELT,IDE,IPR
      TIMEO=TOTALT(JJ+1,1,1)

      DR(1)=DOUT
      IF(JJ.EQ.ILEVM1) GO TO 21

   21 JJ=JJ+1
   11 TTRIP=TRIPT*DEPTH(JJ)/2.+DEPTIN/2.+TIMEIN
      TEQ=TTRIP+TOTRT+BITCST/RIGCST
      CEQ=CEQ+TEQ
      CTRIP=CTRIP+TTRIP
      CROT=TOTRT+CROT
      WRITE(6,6001) FSTART,DEPTH(JJ),TOTRT,TTRIP,TEQ,
     1IDE,IPR
      WRITE(6,6002) DEPTH(JJ)
      SEQ=SEQ+TEQ
      STRIP=STRIP+TTRIP
      SROT=SROT+TOTRT
      CRIG=CROT+CTRIP
      SRIG=SROT+STRIP
      WRITE(6,6010) SROT,STRIP,SRIG,SEQ ,KOUNT
      IF(INDS.EQ.0) WRITE(6,6003) CROT,CTRIP,CRIG,CEQ,KKOUNT
      IF(JJ.EQ.ILEV) GO TO 130
    8 TTRIP=TRIPT*DEPTH(JJ)/2.+DEPTIN/2.
      FSTART=DEPTH(JJ)
      KOUNT=KOUNT+1
      KKOUNT=KKOUNT+1
```

```
C                 MAIN            ...(CONT'D)

      NSEC=1
      WRITE(6,6004)
      [illegible]
      V(1,1)=VIN(JJ)
      R(1,1)=RIN(JJ)
      DE(1)=0.
      PR(1)=0.
      CALL TEST2(V(1,1),R(1,1),TEFF,1,1,IPASS,JJ)
       TOTST=DELT
      [illegible]
      IPR=IFIX(ROUT*[illegible].+0.5)
      WRITE(6,6006) NSEC, DEPTH(JJ),DEPTH(JJ+1),V(1,1),
     R(1,1),DELT,IDC,IPR
      TIMEP=TOTALT(JJ+1,1,1)
      DE(1)=DOUT
      PR(1)=ROUT
        IF(JJ.EQ.ILEVM1) GO TO 21
1     CONTINUE
      READ(5,2)NUSET1,NUSET2
      IF(NUSET1.NE.0) GO TO 999
      IF(NUSET2.NE.0) GO TO 9999
      STOP
```

SUBROUTINE RESULT

```
      SUBROUTINE RESULT(JJ)
C
C     THIS PRINTS OUT A STAGE OF THE PROCESS SHOWING THE
C     OPTIMAL PROCEDURE FOR THAT STAGE FOR EACH OF THE STATE
C     VARIABLE GRID POINTS.
C
      COMMON AR(150),CR(150),DES(150)
      COMMON FOPT(150),FOPTK(150),DEPTH(150)
      COMMON XLIMLO(150,2),XLIMHI(150,2),NOS1(151)
      COMMON [illegible]IN(150),[illegible]I[illegible](150),TOTALT(151,15,15),NOS2(151)
      COMMON XNDIC(15,15),[illegible](15,15),[illegible](15,15),DE(15),[illegible]P(15)
      COMMON ERR,AINC1,AINC2,KEPTI[illegible],CO[illegible](2),TPIPT[illegible],M
      COMMON HOLD,FITCST,[illegible]ICCST,FOPTI[illegible],[illegible]T,NOUT,L[illegible],DELT
C
  101 FORMAT('1',55X,'INITIAL CONDITIONS',/,51X,'DEPTH = '
     1,F6.0,2X,'INTERVAL = ',F6.2,/)
C
      FORMAT(' DEPTH ')
      IND=(NOS2(JJ)-1)/5+1
      KL=NOS1(JJ)
      KM=NOS2(JJ)
      IF(KM.EQ.1) GO TO 11
      KOUNT
C
      WRITE(6,102) (RP(J),J=1,KM)
C
      DO 103 J=1,KL
      KOUNT=KOUN[illegible]
      IF(KOUNT.LE.60) GO TO 105
      WRITE(6,300)
C
      [illegible](6,1[illegible]
      KOUNT=15
      WRITE(6,200) ([illegible](J,K),K=1,KM)
      WRITE(6,201) ([illegible](J,K),K=1,KM)
      WRITE(6,202) DE(J),(TOTALT(JJ,J,K),K=1,KM)
      WRITE(6,203) (XNDIC(J,K),K=1,KM)
      CONTINUE
C
      WRITE(6,112) (RP(J),J=1,KM)
      WRITE(6,10[illegible])
C
      KOUNT=KOUNT+9
      IF(KOUNT.LE.60) GO TO 115
      WRITE(6,300)
      WRITE(6,112) (RP(L),L=1,KM)
C
      KOUNT=15
  115 WRITE(6,210) ([illegible](J,K),K=1,KM)
```

SUBROUTINE RESULT ...(CONT'D)

```
      WRITE(6,211) (R(J,K),K=1,KK)
      WRITE(6,212) DF(J),(TOTALT(JJ,J, [illegible]
      WRITE(6,213) (XMDIC(J,K),K=1,KK)
  113 CONTINUE
      GO TO 7
    6 WRITE(6,122) (PP(J),J=1,KK)
      WRITE(6,109)
      DO 123 J=1,KL
      KOUNT=KOUNT+9
      IF(KOUNT.LE.60) GO TO 125
      WRITE(6,300)
      WRITE(6,122) (PP(L),L =1,KK)
      [illegible]
      [illegible]
  [illegible]
      [illegible]
      WRITE(6,222) DF(J),(TOTALT(JJ,J,K),K=1,KK)
      WRITE(6,223) (XMDIC(J,K),K=1,KK)
  123 CONTINUE
    7 CONTINUE
    3 RETURN
  [illegible]
      GO TO (12,13,14),IND
   [illegible]
      WRITE(6,220) (W(J,1),J=1,KL)
      WRITE(6,221) (R(J,1),J=1,KL)
      WRITE(6,227) (TOTALT(JJ,J,1),J=1,KL)
      [illegible]
      RETURN
   13 WRITE(6,117) (DF(J),J=1,KL)
      WRITE(6,210) (W(J,1),J=1,KL)
      WRITE(6,211) (R(J,1),J=1,KL)
      WRITE(6,217) (TOTALT(JJ,J,1),J=1,KL)
      WRITE(6,213) (XMDIC(J,1),J=1,KL)
      RETURN
   14 WRITE(6,107) (DF(J),J=1,KL)
      [illegible]
      WRITE(6,201) (R(J,1),J=1,KL)
      WRITE(6,207) (TOTALT(JJ,J,1),J=1,KL)
      WRITE(6,203) (XMDIC(J,1),J=1,KL)
      RETURN
  107 FORMAT(62X,'DULLNESS',/,12X,15(3X,F5.3))
  117 FORMAT(62X,'DULLNESS',/,11X,10(7X,F5.3))
  127 FORMAT(62X,'DULLNESS',/, 5X,5(19X,F5.3))
  207 FORMAT(      /,9X,'COST',1X,F6.1,14(2X,F6.1))
  217 FORMAT(      /,9X,'COST',2X,F8.2, 9(4X,F8.2))
  227 FORMAT(      /,9X,'COST',4X,F8.2,4(16X,F8.2))
  102 FORMAT(62X,'BEARINGS',/,12X,15(3X,F5.3))
  112 FORMAT(62X,'BEARINGS',/,11X,10(7X,F5.3))
  122 FORMAT(62X,'BEARINGS',/, 5X,5(19X,F5.3))
```

```
                    SUBROUTINE RESULT ...(CONT'D)

      FORMAT(//,7X,'WEIGHT',1X,F6.1,14(2X,F6.1))
      [illegible]
      FORMAT(//,7X,'WEIGHT',[illegible]X,F[illegible].2,4(1[illegible]X,F[illegible].2))
      FORMAT(/,8X,'SPEED',1X,F6.1,14(2X,F6.1))
      FORMAT(/,8X,'SPEED',2X,F8.[illegible], 9(4X,F[illegible].2))
      FORMAT(/,8X,'SPEED',[illegible]X,F[illegible].2,4(16X,F[illegible].2))
      [illegible]
      [illegible]
      FORMAT(F6.3,/,9X,'COST',[illegible]X,F[illegible].2,4([illegible]X,F[illegible].2))
      FORMAT(/,7X,'POLICY',4X,A1,14(7X,A1))
      FORMAT(/,7X,'POLICY',6X,A1,9(11X,A1))
       FORMAT(/,7X,'POLICY',12X,A1,4(23X,A1))
      END
```

SUBROUTINE LEVEL

```
      SUBROUTINE LEVEL(JJ)
C
C     THIS SUBROUTINE CALCULATES THE OPTIMAL POLICY FOR
C     EACH OF THE STATE VARIABLE GRID POINTS FOR A GIVEN
C     STAGE.
C
      DIMENSION RESP(2)
      COMMON AF(150),CF(150),DFS(150)
      COMMON FORM(150),FORMK(150),DEPTH(151)
      COMMON XLIMLO(150,2),XLIMHI(150,2),NOS1(151)
      COMMON WIN(150),RIN(150),TOTALT(151,15,15),NOS2(151)
      COMMON XNDIC(15,15),W(15,15),R(15,15),DE(15),DN(15)
      COMMON ERR,AINC1,AINC2,REPTIM,CONN(2),TLIMIT,R
     [illegible]
      DATA RESP/'K','R'/
      [illegible]
      DO 2 L=1,15
    2 XNDIC(I,L)=RESP(1)
    1 CONTINUE
      [illegible]
      [illegible]
      NOS2M1=NOS2(JJ)-1
      IF(NOS2M1.EQ.0) NOS2M1=1
      CALL TEST2(XLIMLO(JJ,1),XLIMLO(JJ,2),TEFF,1,1,IANS,JJ)
      [illegible]
     [illegible]
     1+REPTIM
      DO 3 I=1,NOS1M1
      DO 4 L=1,NOS2M1
      IF((I.EQ.1).AND.(L.EQ.1)) GO TO 4
      CALL TEST2(XLIMLO(JJ,1),XLIMLO(JJ,2),TEFF,I,L,IANS,JJ)
      IF(IANS.EQ.1) GO TO 5
      CALL FIND(XLIMLO(JJ,1),XLIMLO(JJ,2),I,L,JJ,TEFF)
      IF(TOTALT(JJ,I,L).GT.REPLAC) GO TO 5
    4 CONTINUE
      GO TO 3
      IF(L.EQ.1) GO TO 6
      DO 7 K=L,NOS2M1
      [illegible]
      W(I,K)=W(1,1)
      R(I,K)=R(1,1)
      XNDIC(I,K)=RESP(2)
    7 CONTINUE
      CONTINUE
      GO TO 10
      DO 8 J=I,NOS1M1
      DO 9 K=L,NOS2M1
      TOTALT(JJ,J,K)=REPLAC
      [illegible]
```

SUBROUTINE LEVEL ...(CONT'D)

```
      R(J,K)=R(1,1)
      XINDIC(J,K)=RESP(2)
      CONTINUE
      CONTINUE
      KL=NOS2(JJ)
      KM=NOS1(JJ)
      IF(KM.EQ.1) GO TO 13
      DO 11 K=1,KL
      [illegible]
      [illegible]
      R(KM,K)=R(1,1)
      XINDIC(KM,K)=RESP(2)
   11 CONTINUE
      IF(KL.EQ.1)  GO TO 14
   13 DO 12 K=1,NOS1 1
      [illegible]
      [illegible]
      R(K,KL)=R(1,1)
      XINDIC(K,KL)=RESP(2)
   12 CONTINUE
   14 AIN(JJ)=R(1,1)
      RIN(JJ)=R(1,1)
      RETURN
      END
```

SUBROUTINE MAPS

```
      SUBROUTINE MAPS(JJ)

C     THIS SUBROUTINE PRODUCES A MAP OF T
C     HE  CALLED.


      DIMENSION XT(16),YV(20),TIME(16,20)
      DIMENSION DT(16,20),AT(16,20)
      COMMON AF(150),CF(150),DFS(150)
      COMMON FOR (150),FOR K(150),J
      COMMON XLIMLO(150,2),XLIMHI(150,2),NOS1(150)
      COMMON NIN(150),NIT(150),TOTALT(2 1,2 ,15),NOS
      COMMON XNUDIC(15,15),(15,15),(15,15),NUD(
      COMMON ENV,AIRC1,AIRC2,NSETIM,COUN(2),TRINT
      COMMON MOLE,BITCST,RIGCST,FOOTIN,ROOFT,ROUT,LAC,DEL
      COMMON IM,XIM
      COMMON IOPT3,IOPT4,IOPT5,IOPT6
      RANGE1=XLIMHI(JJ,1)-XLIMLO(JJ,1)
      IR1=MIN0(IN1,20)
      DO 1 K=1,16
      XT(K)=XLIMLO(JJ,1)+FLOAT(K-1)*RANGE1/1
      DO 2 K=1,IR1
      YV(K)=XLIMLO(JJ,2) +FLOAT(K-1)*RANGE2/FLOAT(IR1-1)

      IF(NOS1(JJ).EQ.1) IOPT3=1

      IF(NOS2J1.EQ.0) NOS2J1=1

      DO 100 I=IOPT3,NOS1J1,IOPT4
      DO 101 L=IOPT5,NOS2J1,IOPT6
      DO 103 K=1,IR1
      DO 102 J=1,16
      CALL TEST2(XT(J),YV(K),TIME(J,K),I,L,IA
      DT(J,K)=DOUT
      AT(J,K)=AOUT
      IF(TIME(J,K).LT.1.E+06) GO TO 102
      TIME(J,K)=0.
  102 CONTINUE
      IND=0
      DO 99 IJK=1,16
   99 IF(AT(IJK,K).GT.1.E5) IND=IND+1
      IF(IND.NE.16) GO TO 103
      IR1=K
      GO TO 300
  103 CONTINUE
```

SUBROUTINE PAPS ...(CONT'D)

```
300 WRITE(6,200) DEPTH(JJ),FOUTINGE(I),[illegible](L)
200 FORMAT('1',5[illegible]X,'INITIAL CONDITIONS',//,32X,'DEPTH = '
   1,F6.[illegible],2X,'INTERVAL = ',F5.[illegible],2X,'[illegible] = ',F6.4,2X
   2,'BEARINGS = ',F6.4,/)
[illegible]
204 FORMAT(64X,'WEIGHT',//,20X,10(1X,F5.2),/,3X,'SPEED')
    KOUNT=0
    DO 104 K=1,IR1
    KOUNT=KOUNT+4
    IF(KOUNT.LE.60) GO TO 106
    KOUNT=4
[illegible]
[illegible]     WRITE(6,201) (TIME(J,K),J=1,10)
    WRITE(6,202) XV(K),(DT(J,K),J=1,10)
    WRITE(6,203) (ST(J,K),J=1,10)
[illegible]     CONTINUE
[illegible]     CONTINUE
[illegible]     CONTINUE
[illegible]     FORMAT(15X,'COST',3X,10(1X,F5.2))
[illegible]
[illegible]     FORMAT(11X,'BEARINGS',1X,10(2X,F5.3),/)
    RETURN
    END
```

```
                  SUBROUTINE FIND

      SUBROUTINE FIND(MATRI[illegible],REVI[illegible],I,L,JJ,TI[illegible])

C     THIS SUBPROGRAM FINDS THE TOTAL MINIMUM COS
C     IT EXAMINES BOTH THE INTERNAL MINIMUM AND THE B[illegible]
C     CONDITION AND TAKES THE LOWER VALUE.
C

      LOGICAL A,B,C,D,E,F
      DIMENSION X1(2),X2(2),XL1(2)
      COMMON AF(150),CF(150),DFS(150)
      COMMON FOR (150),FO[illegible]K(150),DEPTH(151)

      COMMON [illegible]I[illegible](150),[illegible]I (150),TOTALT(15[illegible],15,15),[illegible]OS2([illegible]51)
      COMMON X[illegible]DIC(15,15),W(15,15),P(15,15),[illegible](15),[illegible](15)
      COMMON [illegible],AINC1,AINC2,[illegible]TI[illegible],CO[illegible](2),T[illegible]IPT[illegible],[illegible]
      COMMON HOLL,BITCST,RIGCST,F[illegible]TI[illegible],[illegible],[illegible]OUT,LA[illegible],[illegible]ELT
      SS=1.
      LL=0
      ML=0

      DO 1 J=1,2
    1 X1(J)=X2(J)
      FTEST=TI[illegible]
      CALL DERIV(X1(1),X1(2),XL1,I,L,FTEST,JJ)
      IF(ABS(XL1(1)).LT.1.E-[illegible]
      
      DO 4 J=1,2
      X2(J)=X1(J)-SS*XL1(J)*COEF(J)
      [illegible]
      [illegible]
      CALL TEST2(X2(1),X2(2),FX,I,L,IANS,JJ)
      IF(FX.LT.FTEST) GO TO 6
      IF(SS.LT.ERR) GO TO 100
      SS=SS/2.
      LL=LL+1
      GO TO 5
    6 FTEST=FX
      ML=ML+1
      IF(LL.GT.0) GO TO 30
      SS=SS*2.
      DO 7 J=1,2
      [illegible]
      IF(X2(J).GT.XLIMHI(JJ,J)) X2(J)=XLIMHI(JJ,J)
      IF(X2(J).LT.XLIMLO(JJ,J)) X2(J)=XLIMLO(JJ,J)
      CALL TEST2(X2(1),X2(2),FX,I,L,IANS,JJ)
      IF(FX.LT.FTEST) GO TO 6
      SS=SS/2.
   30 DO [illegible] J=1,2
```

SUBROUTINE FIND ...(CONT'D)

```
      [illegible]                         X1(J)=XLI[illegible]
      [illegible]                         X1(J)=XLIMLO([illegible]
      A=ABS(X1(1)-XLIMHI(JJ,1)).LT.0.1
      [illegible]
      IF(A.AND.B) GO TO 100
      [illegible]
      IF(A.AND.C) GO TO 100
      [illegible]
      IF(B.AND.D) GO TO 100
      [illegible]
      LL=0
      KL=
      GO TO 9
      CONTINUE
      CALL TEST2(XLIMHI(JJ,1),XLIMHI(JJ,2),TMAX,I,L,I[illegible],JJ)
      IF(TMAX.LT.FRACT) GO TO 101
      IF(TMAX.GT.1.E+[illegible]) GO TO 102
      X(I,L)=X1(1)
      Y(I,L)=X1(2)
      [illegible]
      RETURN
 101  [illegible]
      Y(I,L)=XLIMHI(JJ,2)
      [illegible]
      RETURN
      RANGE1=(XLIMHI(JJ,1)-XLIMLO(JJ,1))
      RANGE2=(XLIMHI(JJ,2)-XLI[illegible]
      XGRID=XLIMLO[illegible]
      YGRID=XLIMLO(JJ,2)-RANGE2
      XTEMP=X1(1)
      YTEMP=X1(2)
      TTEMP=[illegible]
1001  [illegible] TEST2 [illegible]
      RANGE1=[illegible]
      [illegible]2=[illegible]
      [illegible]NGE[illegible]
      [illegible]) GO T[illegible]
      E=TTEMP.[illegible]
      IF(E) GO TO 1[illegible]
      IF(RANGE2.LT.0.10) GO[illegible]
1003  IF(TT.GT.1.E+10) GO TO[illegible]
      TTEMP=TT
      XTEMP=XGRID
      YTEMP=YGRID
      XGRID=XGRID+RANG[illegible]
      YGRID=YGRID+RANG[illegible]
      GO TO 1001
1002  XGRID=XGRID-RANGE[illegible]
      [illegible]
      GO TO 1001
```

SUBROUTINE TEST2

```
      SUBROUTINE TEST2(WEIGHT,REVS,TEFF,I,L,IANS,JJ)
C
C     THIS SUBROUTINE CALCULATES THE TOTAL COST TO DATE IF
C     A GIVEN WEIGHT IS USED AT THE CURRENT STAGE.
C
C
      COMMON AF(150),CF(150),DFS(150)
      COMMON FORM(150),FUNK(150),DEPTH(151)
      COMMON XLIMLO(150,2),XLIMHI(150,2),DS1(151),DS2(151)
      COMMON MIN(150),MIN(150),TOTALT(151,15,15)
      COMMON XMDIC(15,15),W(15,15),P(15,15),DE(15),DR(15)
      COMMON ERR,AIMC1,AIMC2,DEPTIM,COEF(2),TRIPT,M
      COMMON HOLE,BITCST,RIGCST,ROTTIM,DOUT,ROUT,LAST,DELT
      COMMON IP,XBAR
       DULLA(D)= 1.0+(6.0+.595125*D)*D
      DULLAI(D)=(1.0+(3.0+.396375*D)*D)*D
      DULLAH(D)=(1.0+(6.0+.396175*D)*D)**0.5
     [illegible]
     .*D)*D)*D
      IANS=0
     [illegible]
      IF(FORM(JJ)-0.5) 1,1,2
    1   ARR=A*REVS**0.75+0.5*REVS*(1.-A)
      GO TO 3
    2 [illegible]
    3 WBAR=XBAR*WEIGHT
      [illegible]
      ROTI=REVS*(1.+.000434*REVS**2)
      XLOAD=21233.07/(1.+.031107*WBAR)**3.005
      GO TO (20,20,30),IP
   20 F1=AF(JJ)*ERATE/ROTI
      F2=F1*CF(JJ)*(WR/R**FORM(JJ))*ARR
      F3=F1*REVS/DFS(JJ)/XLOAD
      GO TO (40,50,30),IP
   40 X=DE(I)+0.2
      C=SIM1(DE(I))+ROTTIM/F2
      DOUT=X-(SIM1(X)-C)/DULLAH(X)
       IF(ABS((DOUT-X)/DOUT).LT.1.E-05) GO TO 5


      IF(DOUT.GT.1.) GO TO 10
       [illegible]
      ROUT=F3*DDIFF+RR(L)
      IF(ROUT.GT.1.) GO TO 10
      CALL INCOL(TPREV,I,L,JJ)
      DELT=F1*DDIFF
      TEFF=TPREV+DELT
      GO TO 11
   10 [illegible]
```

SUBROUTINE TEST2 ...(CONT'D)

```
11 RETURN
50 DOUT=FOOTIN/F2+DE(I)
   IF(DOUT.GT.1.0) GO TO 10
   DDIFF=DULLAI(DOUT)-DULLAI(DE(I))
   [illegible]
   IF(BOUT.GT.1.0) GO TO 10
   CALL INPOL(TPREV,I,L,JJ)
   [illegible]
   TEFF=TPREV+DELT
   RETURN
30 F1=DFS(JJ)*XLOAD/REVS
   BOUT=FOOTIN/(F1*CF(JJ)*(WEAR**POWER(JJ))*ABN)+BB(L)
   [illegible]
   CALL INPOL(TPREV,I,L,JJ)
   DELT=F1*(BOUT-BB(L))
   [illegible]
   RETURN
   END
```

SUBROUTINE DERIV

```
      SUBROUTINE DERIV(WEIGHT,REVS,DTEFF,I,L,TEFF1,JJ)
C
C     THIS SUBROUTINE CALCULATES THE NORMALIZED DERIVATIVES
C     OF COST W.R.T. WEIGHT AND SPEED.
C
C
      DIMENSION DTEFF(2)
      COMMON AF(150),CF(150),DFS(150)
      COMMON FORM(150),FORMK(150),DEPTH(151)
      COMMON XLIMLO(150,2),XLIMHI(150,2),COST1(151)
      COMMON WIN(150),RIN(150),TOTALT(151,15,15),COS2(151)
      COMMON XNDIC(15,15),W(15,15),R(15,15),DE(15),DR(15)
      COMMON ERR,AINC1,AINC2,NEPTIM,CONT(2),TRIPT,[illegible]
      COMMON HOLE,BITCST,RIGCST,ROUTIM,[illegible],[illegible],LAG,[illegible]
      X=WEIGHT-AINC1
      CALL TEST2(X,REVS,TEFF2,I,L,IANS,JJ)
      X=REVS-AINC2
      CALL TEST2(WEIGHT,X,TEFF3,I,L,IANS,JJ)
      [illegible]
      [illegible]
      IF(ABS(DTEFF(1)).LT.1.E-20.AND.ABS(DTEFF(2)).LT.
     11.E-20) GO TO 100
      DEV=(DTEFF(1)**2+DTEFF(2)**2)*[illegible]
      DTEFF(1)=DTEFF(1)/DEV
      DTEFF(2)=DTEFF(2)/DEV
  100 RETURN
      END
```

SUBROUTINE INPOL

```
      SUBROUTINE INPOL(TPREV,I,L,JJ)
C     THIS SUBROUTINE FINDS THE COST UP TO THE PREVIOUS
      STAGE BY INTERPOLATION.

      COMMON AF(150),CF(150),DFG(150)
      COMMON FORM(150),FORMK(150),DEPTH(151)
      COMMON XLIMLO(150,2),XLIMHI(150,2),NOS1(151)
      COMMON WIN(150),XIN(150),TOTALT(151,15,15),NOS2(151)
      COMMON XNDIC(15,15),M(15,15),N(15,15),DE(15),DD(1
      COMMON ERR,AINC1,AINC2,DEPTIN,CON(2),TAINT ,N
      COMMON POLE,BITCST,RIGCST,FOOTIN,DOUT,BOUT,LAS ,DELT
      XNOS1=FLOAT(NOS1(JJ+1)-1)
[illegible]
[illegible]
      IF(M.GT.1.5) GO TO 2
    1 IF(DOUT.LE.(FLOAT(J-[illegible]
      J=J+1
      IF(J.EQ.NOS1(JJ+1)) GO TO 2
      GO TO 1
    2 IF(NOS2(JJ+1).EQ.1) GO TO 3
      IF(BOUT.LE.(FLOAT(K-1)/XNOS2)) GO TO [illegible]
      K=K+1
      IF([illegible].EQ.[illegible])) [illegible]
    3 [illegible]
      J2=J
      IF(M.GT.1.5) J2=J1
[illegible]
      K2=K
      IF(NOS2(JJ+1).EQ.1) K2=K1
      TPREV=0.
      DO 10 J=K1,K2
      TEMP=0.
      DO 11 IJ=J1,J2
      XL1=1.
      IF(M.GT.1.5) GO TO 15
      DO 12 KJ=J1,J2
      IF(IJ.EQ.KJ) GO TO 12
      XL1=XL1*(DOUT*XNOS1-FLOAT(KJ-1))/FLOAT(IJ-KJ)
   12 CONTINUE
[illegible]
   11 CONTINUE
      XL2=1.
      DO 13 KL=K1,K2
      IF(J.EQ.KL) GO TO 13
      XL2=XL2*(BOUT*XNOS2-FLOAT(KL-1))/FLOAT(J-KL)
   13 CONTINUE
```

```
RESULTS OF THE [illegible] OPTIMIZATION
PROCEDURE.  THE GROUPS ARE AS SUGGESTED BY [illegible].


                    GROUP [illegible]

   PIT NO. SECTION      INTERVAL      [illegible]   [illegible]

      1                1400. - 1527.  [illegible]   [illegible]
                       1527. - [illegible]  [illegible]   [illegible]
                       1653. - 178[illegible].  [illegible]   [illegible]
                       1780. - 1907.  [illegible]   [illegible]
                       1907. - 2033.  [illegible]   [illegible]
                       2033. - 2160.  [illegible]   [illegible]
                       2160. - 2287.  [illegible]   [illegible]
                [illegible]. - 2413.  [illegible]   [illegible]
                       2413. - 2540.  [illegible]   [illegible]
                       2540. - 2667.  [illegible]   [illegible]
                       2667. - [illegible]  [illegible]   [illegible]
                       2793. - [illegible]  [illegible]   [illegible]
                       2920. - 3026.  [illegible]   [illegible]
                       3026. - 3133.  [illegible]   [illegible]
                       3133. - 3239.  [illegible]   [illegible]
                       3239. - [illegible]  [illegible]   [illegible]
      5                3345. - 3451.  [illegible]   [illegible]
                       3451. - 3558.  [illegible]   [illegible]
                       3558. - 3664.  [illegible]   [illegible]
                       3664. - 377[illegible].  [illegible]   [illegible]
                [illegible]. - 3851.  [illegible]   [illegible]
                [illegible]. - 3933.  [illegible]   [illegible]
                [illegible]. - 4014.  [illegible]   [illegible]
                [illegible]. - 4095.  [illegible]   [illegible]
                [illegible]. - 417[illegible].  [illegible]   [illegible]
                [illegible]6. - [illegible]  [illegible]   [illegible]
                  [illegible]259. - [illegible]  [illegible]   [illegible]
                  [illegible]339. - [illegible]  [illegible]   [illegible]
                  [illegible]420. - [illegible]  [illegible]   [illegible]
                  [illegible]501. - [illegible]  [illegible]   [illegible]
                  [illegible]582. - [illegible]  [illegible]   [illegible]
                  [illegible]664. - [illegible]  [illegible]   [illegible]
                  [illegible]. - [illegible]  [illegible]   [illegible]
                [illegible]. - [illegible]  [illegible]   [illegible]
                [illegible]. - [illegible]  [illegible]   [illegible]
                [illegible]. - 5[illegible]7.  [illegible]   [illegible]
                [illegible]. - 5178.  [illegible]   [illegible]
                [illegible]. - 528[illegible].  [illegible]   [illegible]
                [illegible]. - 5389.  [illegible]   [illegible]
                [illegible]. - 5495.  [illegible]   [illegible]
                [illegible]. - 5601.  [illegible]   [illegible]
                [illegible]. - 5708.  [illegible]   [illegible]
                [illegible]. - 5814.  [illegible]   [illegible]
                [illegible]. - 592[illegible].  [illegible]   [illegible]
```

```
BIT NO. SECTION      INTERVAL          [illegible]  [illegible]

 12              5920. -   6025.   56.       101.
                 [illegible] -   6133.   59.       [illegible]
                 6133. -   6229.   [illegible]       [illegible]
                 [illegible] -   [illegible]   [illegible]       [illegible]
                 [illegible] -   6451.   56.       [illegible]
                 6451. -   [illegible]   [illegible]       [illegible]
                 [illegible] -   [illegible]   60.       [illegible]
                 [illegible] -   [illegible]   [illegible]       [illegible]

                 GROUP NO.  2

 [illegible]     SECTION      INTERVAL

 [illegible]        1    [illegible] -   [illegible]

                 GROUP NO.  3

                 SECTION      INTERVAL

                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   7112.
                 [illegible] -   7212.
                 7212. -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]
                 [illegible] -   [illegible]

                 GROUP NO.  4

 [illegible]                  INTERVAL

 [illegible]     [illegible] -   [illegible]
```

```
                     GROUP NO.  5

[illegible] NO. SECTION     INTERVAL     HEIGHT   SPEED

      1          1      8800.  -  9020.    60.      100.

                     GROUP NO.  6

[illegible]    SECTIO       INTERVAL     [illegible]

      1   [illegible]   9020.  -  9050.  [illegible]

                     GROUP NO.  7

[illegible]                 INTERVAL     HEIGHT   SPEED

      1          1   [illegible]  -  [illegible]  54.   [illegible]
                 2      9115.  -  [illegible]  60.  [illegible]
                 3      9165.  -  9215.    60.  [illegible]
                 4      9215.  -  9270.    60.  [illegible]
                 5      9270.  -  9271.  [illegible]
                 6   [illegible]  -  9475.  60.  [illegible]
                 1   [illegible]  -  9574.  55.  [illegible]
                 2   [illegible]  -  9675.  [illegible]
                 3   [illegible]  -  9775.  [illegible]
                 4   [illegible]  -  9875.  60.  [illegible]
                 5      9875.  -  [illegible]  [illegible]
                 6      9975.  -  10075.  [illegible]

                     [illegible]

[illegible]  SECTION  [illegible]

      1          1   [illegible]  -  10152.  [illegible]
                 2   [illegible]
                 3   [illegible]
      2          1   [illegible]
                 2     10527.  -  10655.  [illegible]
                 3     10656.  -  [illegible]          3.
                 4   [illegible]                       2.
                 5     10711.  -  [illegible]          1.
      3          1     10813.  -  10844.               2.
                 2     10844.  -  10875.               2.
                 3     10875.  -  [illegible]
                 4     10906.  -  10936.
  [illegible]    1     10936.  -  [illegible]
                 2     10967.  -  [illegible]
                 3     [illegible]  -  [illegible]
                 4     11031.  -  [illegible]
  [illegible]    1     [illegible]  -  [illegible]
                 2     11094.  -  [illegible]
                 3     11125.  -  [illegible]
```

```
                       INTERVAL

             4    11157. - [illegible]                   55.
             1    111[illegible]. - [illegible]          21.
                  [illegible] - [illegible]              33.
             3    11250. - 112[illegible]2.              [illegible]4.
             4    [illegible] - [illegible]              55.

                     [illegible] NO. 9

                       INTERVAL    WEIGHT

   1              [illegible] - [illegible]

                       INTERVAL                SPEED

                  11680. - 11735.                        55.
                  11738. - [illegible]                   70.
                  11795. - [illegible]                   70.
                  11853. - [illegible]                   70.
                  11911. - [illegible]                   35.
                  [illegible] - 12026.                   65.
                  [illegible] - 12084.                   70.
                  120[illegible]4. - 121[illegible].     70.
   3              [illegible] - 12256.                   36.
                  [illegible] - 1231[illegible].         31.
                  12312. - 123[illegible].               70.
             4    [illegible] - 124[illegible]4.         70.
                  [illegible] - 125[illegible].          35.
                  12540. - 126[illegible].               62.
                  12616. - 126[illegible]2.              70.
                  [illegible] - 12766.                   70.
   5              [illegible] - 12844.                   31.
                  [illegible]2844. - 129[illegible].     [illegible]
                  12920. - 1297[illegible].              [illegible]
                  [illegible] - 130[illegible].          [illegible]
                  130[illegible]. - 13098.               [illegible]
                  130[illegible]. - [illegible]31[illegible]7.   [illegible]
                  131[illegible]7. - [illegible]3216.    [illegible]
                  13216. - 132[illegible].               [illegible]
                  13275. - [illegible]                   [illegible]
                  13334. - [illegible]                   [illegible]
                  13342. - [illegible]                   [illegible]
                  1[illegible]4[illegible]. - [illegible]   [illegible]

CUMULATIVE ROTATING TIME          [illegible]    RIG HOURS
CUMULATIVE TRIP TIME              [illegible]    RIG HOURS
CUMULATIVE RIG TIME               [illegible]    RIG HOURS
CUMULATIVE COST                   [illegible]    RIG HOURS
CUMULATIVE BIT COUNT
```

B29943